Optimización de redes con arcos de baja confiabilidad adicionando arcos redundantes

por

Paola Andrea Hernández Ramírez

Tesis sometida en cumplimiento parcial
de los requisitos para el grado de

MAESTRO EN CIENCIAS
en
Ingeniería Industrial

UNIVERSIDAD DE PUERTO RICO
RECINTO UNIVERSITARIO DE MAYAGÜEZ
2007

Aprobado por:

_____	_____
Viviana Cesaní, Ph.D. Miembro, Comité Graduado	Fecha
_____	_____
Mercedes Ferrer, M.S. Miembro, Comité Graduado	Fecha
_____	_____
Noel Artiles, Ph.D. Presidente, Comité Graduado	Fecha
_____	_____
Freya M. Toledo, M.S. Representante, Estudios Graduados	Fecha
_____	_____
Agustín Rullán, Ph.D. Director del Departamento	Fecha

Abstract

This thesis develops a methodology to optimize the reliability of a complex network whose nodes are not necessarily in series or in parallel and their arcs have low known reliability. Network reliability is improved by using redundant arcs with a budgetary restriction.

Two heuristic optimization algorithms are used and compared to solve this redundancy allocation problem: a genetic algorithm (AG) and an algorithm proposed by the doctor Noel Artiles and developed by the author based on sequential integer linear programming (PLES). Since the exact evaluation of the objective problem is very difficult, it is evaluated by Monte Carlo simulation.

Two factorial designs were developed to compare the maximum network reliability (R_{RT}) and the execution time (t_e) of the heuristics optimization, AG and PLES. In conclusion, the maximum network reliability only was affected by the network size (n), therefore both heuristics maximize the network reliability (R_{RT}) with the same efficiency. Furthermore, the experiment results shown that the proposed heuristic PLES is a faster algorithm than AG for all the network sizes.

Resumen

En esta tesis se desarrolla una metodología para optimizar la confiabilidad de una red compleja cuyos nodos no pueden ser agrupados en serie y paralelo y sus arcos tienen confiabilidades bajas conocidas. La confiabilidad de la red es mejorada adicionando arcos en redundancia con una restricción de presupuesto.

Dos heurísticos de optimización son usados y comparados para resolver este problema de asignación de redundancia: un algoritmo genético (AG) y un algoritmos propuesto por el doctor Noel Artiles y desarrollado por la autora basado en programación lineal entera secuencia (PLES). La evaluación exacta de la función objetivo de este problema es muy difícil, por lo tanto ésta es evaluada utilizando simulación de Monte Carlo.

Se desarrollaron dos experimentos factoriales con el fin de comparar la confiabilidad máxima alcanzada (R_{RT}) y el tiempo de ejecución (t_e) de los dos heurísticos de optimización, AG vs. PLES. A partir de estos experimentos se concluyó que la confiabilidad máxima alcanzada sólo se veía afectada por el tamaño de la red (n) y por lo tanto los dos heurísticos maximizan la confiabilidad de la red (R_{RT}) con igual eficiencia. Adicionalmente, los resultados de los experimentos también permitieron concluir que el heurístico propuesto PLES resultó ser un algoritmo significativamente más rápido que AG para todos los tamaños de red.

A mi papá por ser el hombre de mi vida y mi mayor ejemplo, a mi mamá por ser una mujer íntegra y luchadora digna de ser admirada y seguida, a mi hermana por "obvio si" motivarme a intentar ser un buen ejemplo y brindarme su loca ternura, a mi hermanito por ser la razón y el motor de mi existencia y a mi sobrinita María José que viene en camino y sé que nos va a unir como familia pese a la distancia. A Andrés, mi amor y compañero por ser mi apoyo y siempre estar a mi lado para ofrecerme su soporte y cariño.

Agradecimientos

Deseo agradecer a mi familia, mis padres y hermanos quienes siempre han confiado en mis capacidades y pese a la distancia me han ofrecido su apoyo y me han hecho llegar sus energías para alcanzar mis metas. A mi particular y gran familia en Colombia, mi abuelita, tíos y primos por motivarme día tras día con su amor y manifestar el gran orgullo que profesan por cada uno de mis pasos positivos.

Gracias a Andrés, por ser mi principal compañía durante estos cuatro años juntos, por enseñarme a ser un poco más disciplinada y a querer mejorar cada día pues estoy muy lejos de ser perfecta, por buscar siempre soluciones a mis problemas y estar a cada minuto a mi lado en el desarrollo de esta investigación.

Quiero agradecer de todo corazón el apoyo y orientación que me brindó durante mi período de estudio en la maestría y el desarrollo de esta tesis a mi consejero y maestro el doctor Noel Artiles, sin duda alguna es un gran académico y me siento orgullosa y privilegiada de haber contado con su guía y soporte para sacar adelante esta importante prueba.

Un especial agradecimiento a mis profesores de la maestría quienes con su enseñanza y conocimiento me ayudaron en la elaboración de esta investigación y su inigualable tarea de docencia me motivó para desear continuar con mis estudios de doctorado en búsqueda de alcanzar el sueño de ser maestra como ellos. A la doctora Viviana Cesaní a quien además de agradecerle su participación activa en mi comité graduado le doy gracias por brindarme siempre un saludo amable y una especial sonrisa, al doctor David González quien con su alegría y pasión por la enseñanza despertó en mi el interés por la estadística y el diseño de experimentos, al doctor Pedro Resto quien siempre me brindó sus consejos sinceros y se preocupó por mi bienestar y futuro y a la doctora Sonia Bartolomei a quien admiro como profesora y le agradezco la oportunidad de haber trabajado juntas en mi tarea como instructora del laboratorio de diseño de planta. Extiendo mis agradecimientos a la doctora Lourdes Rosario quien me ofreció su mano amiga desde mi llegada al recinto y confió en mis capacidades para dictar el laboratorio de manufactura en Ingeniería Mecánica.

Gracias a Mayra y Laura por brindarme su colaboración en todo momento y siempre ser tan especiales y amables con los estudiantes. A Edwin Morales por su ayuda en lo relacionado con el centro de cómputo y los equipos, por querer a los latinos y brindarme su amistad y cariño paternal.

A mis compañeros y amigos de maestría, en especial a Ana María quien con su alegría e incondicional amistad hizo que la oficina de graduados fuera más colorida y amañadora, a Edwincito, un hombre intachable a quien siempre voy a recordar como mi colega más admirado, a quien le agradezco su apoyo con esta investigación en la solución de un sinnúmero de dudas en programación, probabilidad y estadística, a Vera, Vladimir, Liliana, Andrés, Verónica, Alejandro y Dennis, por ser mis amigos y siempre estar a mi lado, a Victoria y Mericia por estar conmigo en este último año y animarme en la etapa más difícil, jamás me dejaron sola y nos reímos muchísimo juntas. A Luis y Fabián por su soporte en programación y su disponibilidad siempre amable para ayudarme cuando Matlab o los equipos eran el motivo de mi bloqueo.

A Rafael, por ser el mejor de los amigos siempre preocupado por mi futuro y éxito, por enseñarme que esta isla tiene gente bella con hermosos sentimientos y por regalarme un pedacito de su gran corazón.

Gracias Puerto Rico, por acogerme estos tres años de mi vida y suministrarme las herramientas académicas y económicas para obtener mi título de maestría.

Tabla de Contenido

Página

Lista de Tablas

Tablas

Página

Lista de Figuras

Figuras

Página

1 Introducción

1.1 Motivación y Propósito General

Muchos sistemas, tales como redes de telecomunicaciones, redes de transporte y redes de energía, pueden ser representados por diagramas de redes. Una red consiste de nodos y conexiones o arcos. Usualmente ambos, nodos y arcos, son propensos a fallas. La confiabilidad de la red es la probabilidad de que una red (sistema) funcione satisfactoriamente por un período de tiempo establecido bajo ciertas condiciones y restricciones. El funcionamiento de una red puede considerarse satisfactorio de muchas maneras; por ejemplo, todos los nodos de la red deben estar conectados, un nodo específico debe estar conectado con los demás nodos de la red o dos nodos específicos deben estar conectados, como es el caso analizado en esta tesis.

El problema de optimización de la confiabilidad de redes con probabilidad de falla empleando arcos redundantes ha sido estudiado por varios autores que han desarrollado algoritmos que resuelven este problema para redes que denominan de gran tamaño con estructuras en serie y paralelo, presentando ejemplos para redes hasta con 50 nodos (Prasad, Aneja y Fair (1991)). Existen algunas publicaciones que muestran el trabajo con redes cuyos elementos no están en una configuración serie-paralelo, presentando ejemplos y resultados para redes con máximo 20 nodos y 55 arcos (Dengiz, Antiparmak y Smith (1997)). Estas publicaciones no discuten los algoritmos usados ni su efectividad. La mayoría de las investigaciones presentan el trabajo de optimización de redes adicionando únicamente arcos que se encuentran en redundancia, es decir, arcos paralelos a arcos ya existentes.

En este trabajo se optimiza una red compleja considerando arcos con probabilidades altas de falla (las cuales se presumen conocidas) mediante la adición de arcos, algunos de ellos redundantes, con un presupuesto de inversión limitado. Una red compleja es aquella que no puede ser dividida en grupos de nodos en serie y paralelo. La redundancia es definida como el uso de componentes similares funcionalmente (no necesariamente idénticos) juntos en un diseño de forma que si uno de los componentes falla la parte redundante está disponible para desempeñar la función requerida sin que la red experimente fallas.

Las redes complejas a evaluar son de tipo dirigido (unidireccional) con flujo desde el nodo de menor índice hacia el nodo de mayor índice (Figura 1).

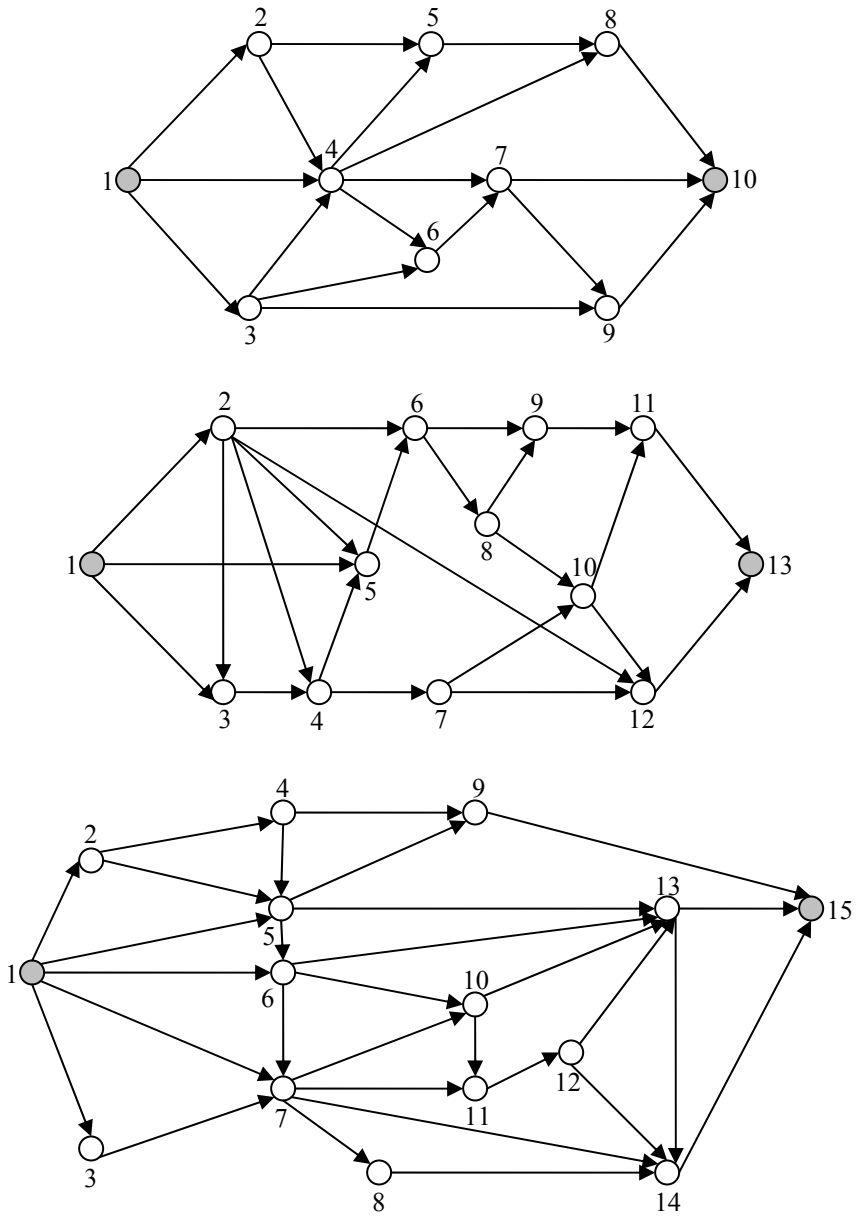


Figura 1. Ejemplos de redes complejas unidireccionales con flujo del nodo de menor índice hacia el nodo de mayor índice (Kuo, Lu y Yeh (1999))

La evaluación de la función objetivo de este problema de optimización no lineal entero, maximizar la confiabilidad de una red compleja, se dificulta, dado el número de variables y la complejidad de la expresión, a medida que las redes crecen en tamaño (Coit y Smith (1995, 1996), Kuo y Prasad (2000)). Para determinar la ubicación óptima de los arcos se utilizarán y

compararán dos metodologías: un algoritmo genético (AG) y un algoritmo desarrollado basado en programación lineal entera secuencial (PLES).

1.2 Objetivos

Los objetivos de esta tesis son:

- Hacer una revisión de literatura sobre la optimización de la confiabilidad en redes mediante la ubicación óptima de redundancias y que han incluido el uso de algoritmos genéticos y otros heurísticos para resolver este problema. En esta investigación, se considera como confiabilidad de las redes a la probabilidad de que la red funcione dado que el nodo origen se encuentra conectado con el nodo final.
- Formular un modelo de optimización cuya función objetivo es la maximización de la confiabilidad de una red compleja mediante la adición de arcos en redundancia con una restricción de presupuesto de inversión.
- Establecer una nueva metodología (PLES) y optimizar la confiabilidad de redes complejas. Determinar y describir detalladamente el procedimiento y parámetros del heurístico a proponerse (PLES).
- Comparar el desempeño del algoritmo propuesto (PLES) con algoritmos existentes (AG), para determinar si el heurístico propuesto es una mejor alternativa para resolver el problema de optimización de la confiabilidad de redes complejas adicionando arcos en redundancia.

1.3 Revisión de Literatura

En problemas de diseño óptimo de confiabilidad, la función objetivo puede ser maximización de la confiabilidad de la red o maximización de la disponibilidad de la red. Las restricciones pueden incluir limitantes de presupuesto, requerimientos de confiabilidad y otras consideraciones. Las variables de decisión pueden ser valores de componentes de confiabilidad, número de redundancias o arreglos de componentes conocidos.

Los problemas de diseño de confiabilidad pueden ser formulados como problemas de optimización, y se han utilizado varios algoritmos de optimización para resolverlos. El mayor énfasis del trabajo reciente en el área de optimización de confiabilidad de redes es el desarrollo de métodos y algoritmos heurísticos y metaheurísticos para problemas de asignación de redundancia.

Kuo y Zuo (2003) establecen que la literatura sobre métodos de optimización de confiabilidad puede ser clasificada en siete categorías:

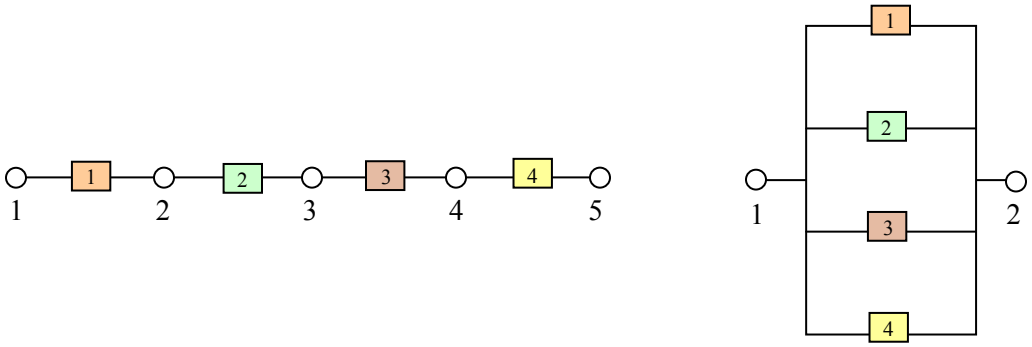
1. Heurísticos para asignación de redundancia.
2. Algoritmos metaheurísticos para asignación de redundancia.
3. Algoritmos exactos de asignación de redundancia o asignación de confiabilidad
4. Heurísticos para asignación de confiabilidad-redundancia.
5. Sistemas multiobjetivos de optimización de confiabilidad.
6. Asignación óptima de componentes intercambiables.
7. Otros: incluyendo descomposición y minimización de función de esfuerzo.

1.3.1 Confiabilidad de Redes

Kuo y Zuo (2003) indican que las estructuras comúnmente utilizadas en sistemas de confiabilidad son las estructuras en serie y paralelo (Figura 2). Cada componente en una red en serie es esencial para el funcionamiento de la red completa. La confiabilidad de las redes en serie está afectada significativamente por el número de componentes en la red. A medida que la red tenga más componentes en serie su confiabilidad es menor. Como resultado, los investigadores y diseñadores han enfocado gran parte de sus esfuerzos en reducir el número de componentes que están conectados en serie dentro de una red. Un método alternativo para incrementar la confiabilidad de una red en serie es aplicar redundancia a nivel de componentes. Estos producirían subsistemas en paralelo.

Una red con estructura en paralelo requiere al menos un componente para trabajar. Mientras el componente esté trabajando, la red funciona. Todos los demás componentes son llamados componentes redundantes. La confiabilidad de la red es mayor que la del mejor componente. De manera opuesta a la red en serie, a medida que la red tenga más componentes en paralelo su confiabilidad es mayor.

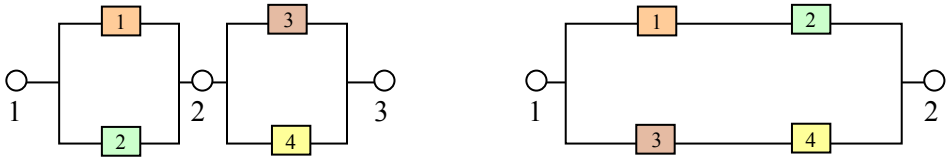
Realmente, no importa cuán baja sea la confiabilidad del componente, se puede incrementar la confiabilidad de la red empleando redundancia. Sin embargo, existen otros elementos que hay que considerar como los costos de inversión que limitan el uso de componentes redundantes en exceso.



a. Serie b. Paralelo

Figura 2. Ejemplos de una red en serie y una red en paralelo

La redundancia es más efectiva cuando es aplicada al nivel más bajo en una red jerárquica (Figura 3).



a. Redundancia a nivel de componente b. Redundancia a nivel de subsistema

Figura 3. Diferentes filosofías del uso de redundancia

En los estudios predomina la presunción que sólo uno de los dos elementos de la red falla, o los nodos o los arcos (Kuo y Zuo, 2003). Con esta suposición, una red puede ser trabajada de un modo similar a un diagrama de bloque de confiabilidad bajo el postulado que hay un solo recurso y un solo fracaso en la red. La operación exitosa de una red puede ser definida de manera diferente para varios propósitos. Una red puede ser definida funcionando si dos nodos específicos se encuentran comunicados. En este caso, se dice que se está tratando con un problema de confiabilidad de red de dos terminales. Si se define que la red funciona adecuadamente si k nodos son capaces de comunicarse con otro, entonces se está tratando con un problema de confiabilidad de red de k terminales.

Existen diferentes métodos para evaluar la confiabilidad de una red: reducciones paralelas y series, el método de descomposición, el método de inclusión-exclusión, el método de la suma de productos separados, el método de transformaciones "delta-estrella" y "estrella-delta", y el método de estructuras de cadenas de Markov. La mayoría de estos métodos requieren del conocimiento de las rutas mínimas o cortes mínimos.

En una red compleja pueden identificarse subsistemas en paralelo y serie y aplicar reducciones. En una reducción en serie, un subsistema en serie con *n* componentes es reemplazado con un supercomponente cuya confiabilidad es igual al producto de las confiabilidades de los componentes en el subsistema. En una reducción en paralelo, un subsistema en paralelo con *n* componentes es reemplazado con un supercomponente cuya confiabilidad es igual a uno menos el producto de la falta de confiabilidad de los componentes en el subsistema.

En la evaluación de la confiabilidad de una red, las reducciones en paralelo y serie siempre son aplicadas primero. Cuando no son posibles más reducciones, otras técnicas son usadas.

El método de descomposición está basado en el concepto de la probabilidad condicional como se presenta en la siguiente ecuación:

$$\Pr(\text{red trabaje}) = \Pr(\text{componente i trabaje})\Pr(\text{red trabaje} \mid \text{componente i trabaje}) + \Pr(\text{componente i falle})\Pr(\text{red trabaje} \mid \text{componente i falle})$$

La eficiencia de éste método depende de la facilidad de evaluación de las probabilidades condicionales. Esto significa que la selección del componente a ser descompuesto tiene un rol importante en la eficiencia del método.

El método de inclusión-exclusión (IE) es un método clásico para producir una expresión de confiabilidad de una red general usando rutas mínimas o cortes mínimos. Está basado en el principio IE para la evaluación de la unión de varios eventos. El método IE, también conocido como el teorema de Poincaré y Sylvester's, provee los límites superior e inferior sucesivos de la confiabilidad de la red por medio de las desigualdades de Bonferroni que convergen a la confiabilidad exacta de la red cuando todas las sumatorias de probabilidades de que las rutas mínimas de la red funcionen simultáneamente hayan sido incluidas.

Kuo y Zuo (2003) dicen que el método de la suma de productos separados (SPS) fue presentado por primera vez en 1973 por Fratta y Montanari. El método SPS usa rutas o cortes mínimos para evaluar la probabilidad de la unión de varios eventos. La unión de las rutas

mínimas puede ser expresada por la función lógica de la red. Esta función lógica puede ser expresada como la unión de varios términos. Si los términos de la función lógica están separados, entonces hay una correspondencia uno a uno entre la expresión de la función lógica de la red y la medida de su confiabilidad. Por lo tanto, el objetivo de este método es expresar la función lógica de la red como una unión de términos separados. Cada término separado es un producto de los eventos correspondientes a que componentes individuales funcionen o fallen. El método SPS difiere del método IE en los signos (positivo o negativo) de los términos en la fórmula de confiabilidad de la red. Con el método IE, los signos de los términos alternan entre más y menos con el signo de suma denotando los subsistemas cuya probabilidad es adicionada a la fórmula de confiabilidad y el signo de resta denotando los subsistemas cuya probabilidad substrae de la fórmula de confiabilidad debido a la cuenta doble en las operaciones anteriores de la inclusión. Con el método SPS, a diferencia del método IE, todos los términos tienen signo positivo y no hay doble conteo; por lo tanto, para cualquier sistema la fórmula SPS es más corta que la fórmula IE por contener menos términos en la expresión.

El método de transformaciones "delta-estrella" y "estrella-delta" permite hacer reducciones adicionales en paralelo y/o en serie mediante transformaciones en la estructura de la red. Se presume que los nodos de la red son perfectos mientras los arcos son propensos a falla. Los componentes propensos a falla son independientes. En una transformación "delta-estrella" la estructura delta (Figura 4.a.) es reemplazada con la estructura estrella (Figura 4.b.) y las confiabilidades de los componentes propensos a falla en la estructura estrella son expresados como funciones de las confiabilidades de los componentes propensos a falla de la estructura delta original.

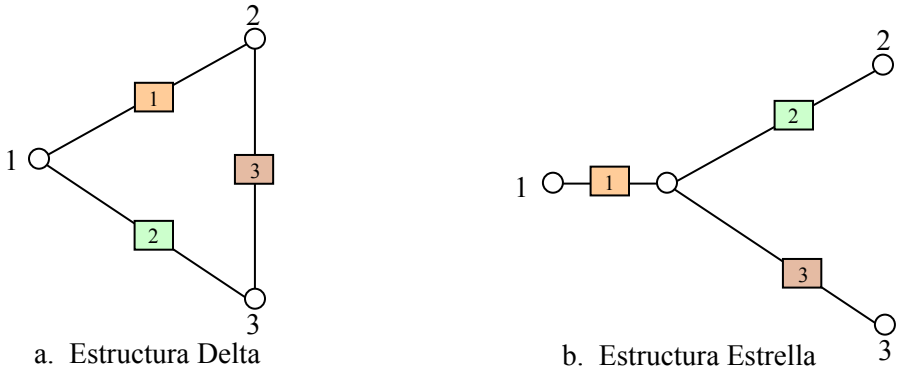


Figura 4. Ejemplo de estructuras delta (a) y estrella (b)

Una trasformación “delta-estrella” será realizada si simplifica la red original. Una transformación “delta-estrella” permite hacer otras transformaciones “delta-estrella”, transformaciones “estrella-delta”, y posibles reducciones en paralelo y en serie. En una transformación “estrella-delta” la estructura estrella es reemplazada con la estructura delta y las confiabilidades de los componentes de la estructura delta son expresados como funciones de las confiabilidades de los componentes de la estructura estrella original. De igual manera que la transformación “delta-estrella”, la transformación “estrella-delta” es realizada sólo si simplifica la red original. Una transformación “estrella-delta” permite hacer otras transformaciones “estrella-delta”, transformaciones “delta-estrella”, y posibles reducciones en paralelo y en serie. La transformación que se realizará debe producir una estructura que sea equivalente a la estructura original. Para que una estructura delta sea equivalente a una estructura estrella en la evaluación de la confiabilidad de la red, se deben conservar las probabilidades punto a punto.

Kuo y Zuo (2003) explican la metodología de cadenas de Markov haciendo referencia a que su primer uso en el análisis de la estructura de confiabilidad de un sistema fue presentado por Chao y Lin (1984). La estructura analizada por estos autores es la llamada estructura F: “*k*-componentes consecutivos de *n*”. La estructura F supone que *n* componentes están linealmente conectados de tal manera que el sistema falla si y sólo si al menos *k* componentes consecutivos fallan. Un vector binario de dimensión *k* es usado para representar el estado de la cadena de Markov en el sistema F. La cadena de Markov tiene 2^k posibles estados y, como resultado, es útil sólo para valores pequeños de *k*. En investigaciones siguientes se utilizó un escalar para representar los estados de la cadena de Markov el cual hace de la técnica una herramienta eficiente para la evaluación de la confiabilidad en este tipo de sistemas. Más adelante, estandarizaron esta aproximación creando un marco general para el análisis de varios tipos de estructuras de sistemas. Las estructuras de sistemas que pueden ser representadas por una cadena de Markov son llamados “sistemas conectados linealmente”.

1.3.2 Optimización de Redes

Aneja, Chandrasekaran y Nair (2004) enfocan su investigación en redes cuyos componentes provienen de conjuntos de selección discreta. En un conjunto de selección, las alternativas tienen costos incrementales con incremento de confiabilidad. Su objetivo es asegurar costos mínimos para conseguir una confiabilidad específica en las redes que se consideran. Desarrollan

formulaciones y algoritmos alternativos basados en aproximaciones de programación dinámica, y estos son generalizados para redes S/P-reducibles. Los autores consideran que el algoritmo desarrollado representa una herramienta efectiva para el diseño de redes con un costo mínimo.

Prasad, Aneja y Nair (1991) desarrollan un heurístico de aproximación para asignar de manera óptima componentes a una red de tipo paralelo-serie. Una red paralelo-serie consiste en un conjunto de rutas en paralelo y cada conjunto de rutas está conformado por componentes posicionados en serie. De manera opuesta, una red serie-paralelo tiene conjuntos de cortes en serie y cada conjunto de cortes tiene componentes en paralelo.

Los autores presentan un algoritmo heurístico que requiere resolver k(k+1)/2 problemas asignados para una solución heurística de cruce de un problema de maximización de confiabilidad sujeto a la restricción de número de componentes. En la primera iteración, se selecciona un conjunto de rutas las cuales pueden tener una alta confiabilidad cuando el número de componentes apropiados es seleccionado del conjunto M (conjunto de posibles componentes) y asignado de manera óptima. Después de eliminar del conjunto M los componentes que son asignados al conjunto de rutas seleccionado en función de maximizar su confiabilidad, se selecciona otro conjunto de rutas desde los componentes sobrantes siguiendo el mismo criterio. Este proceso se repite hasta que finalmente se obtiene un orden de conjuntos de rutas y asignación de componentes para todas las posiciones en cada conjunto. Esta asignación se considera óptima heurísticamente.

El objetivo de la investigación de Prasad, Aneja y Fair (1991) es evaluar qué o con qué frecuencia se genera la solución óptima por el heurístico propuesto. Para esto, comparan la confiabilidad obtenida por el heurístico con la confiabilidad óptima resultante de la enumeración completa. Pese a que se establece que no hay un método exacto disponible para calcular la confiabilidad óptima, se opta por trabajar con una enumeración completa que involucra un gran trabajo computacional para sistemas grandes. Los resultados numéricos obtenidos para 20 problemas generados aleatoriamente, con sistemas desde 4 hasta 50 componentes, apoyan la aplicación del método heurístico a partir de las estadísticas recopiladas que indican que el 15% de las confiabilidades obtenidas con el heurístico no coincidieron con las confiabilidades exactas del sistema y su desviación relativa con respecto a los valores exactos fue del 3.6%.

Otro heurístico que presenta la literatura, enfocado a resolver problemas de optimización de redundancia restringida en redes complejas, es el desarrollado por Kim y Yum (1993).

El método propuesto permite "excursiones" sobre una región limitada de soluciones no factibles. La conclusión establecida por los autores es que los resultados presentados por su heurístico son consistentemente mejores en términos de calidad de solución que otros métodos heurísticos publicados como lo es el desarrollado por Kohda e Inoue (1982) una década antes y el heurístico de recocido simulado.

El algoritmo de Kim y Yum (1993) consiste de una serie de iteraciones que siempre empiezan con la mejor solución actual (x^c) (sea x el conjunto de x_i que corresponden al número de componentes en el subsistema i), hacen una "excursión" en la región no factible limitada (BIFR), y eventualmente:

Caso 1: retorna a la región factible (FR) con una solución x^f, o

Caso 2: termina con una solución x^t que se encuentra fuera de la región BIFR.

En el caso 2, todo el algoritmo se detiene, y, $x^* = x^c$. En el caso 1, un intento nuevo es hecho para mejorar x^f de acuerdo al método descrito y luego x^c es comparado con x^f (posiblemente mejorado). Si la confiabilidad de x^f es mayor que la de x^c, entonces x^c es reemplazado por x^f y empieza una nueva iteración.

Otras reglas y consideraciones complementan el algoritmo descrito y mediante la solución de problemas de redes complejas (Figura 5) con un máximo de 15 nodos conociendo sus costos y confiabilidades asociadas, concluyen que en términos de calidad de solución el heurístico propuesto es mejor que otros presentados, no obstante, en términos de tiempo de solución, métodos como el de Shi (1987) requiere menor cantidad de tiempo de cómputo.

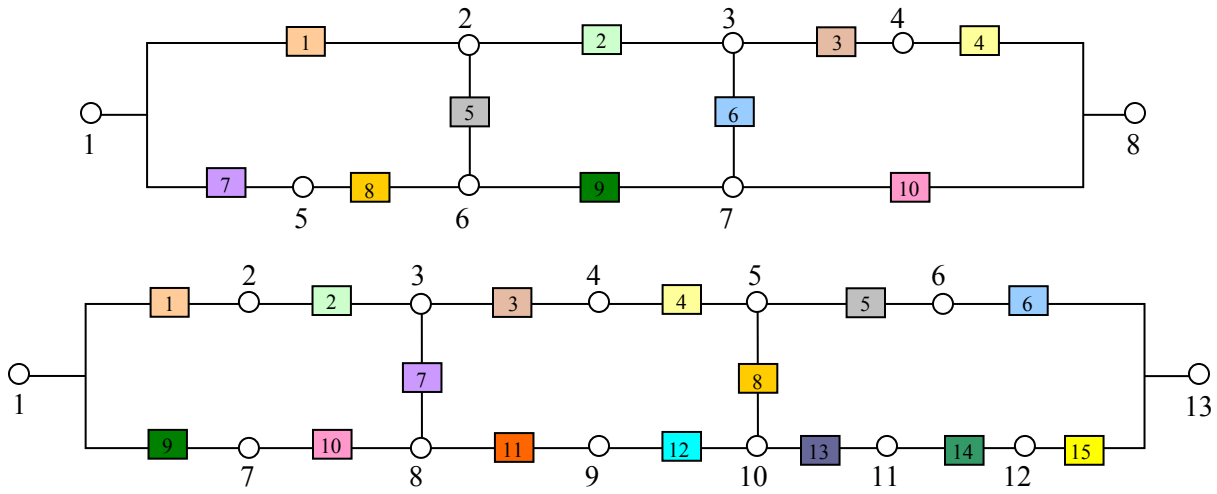


Figura 5. Ejemplos de redes complejas

Jacobson y Sant (1996) concentran su investigación, presentada en el Simposio Anual de Confiabilidad, en el camino para mejorar la confiabilidad de redes a través de asignación óptima de redundancia al nivel de subsistemas, mientras las redes están sujetas a algunas restricciones. Por ejemplo, en problemas de energía eléctrica, generación de redes redundantes proveerían almacenamiento de seguridad de energía para compensar posibles situaciones de sobrecarga. Cuando un computador en un sistema de control de tráfico aéreo falla, la red es operada con un sistema de seguridad que contiene la misma información que el primario. Se han hecho excelentes trabajos en el área de asignación óptima de redundancia.

Jacobson y Sant (1996) combinan el método de búsqueda simplex con una aproximación heurística desarrollada por Aggarwal (1975) como una técnica de optimización para resolver el problema de confiabilidad. El procedimiento se describe como simple y eficiente; un subsistema de confiabilidad es inicialmente asumido y el número óptimo de redundancias es determinado por la técnica heurística. Un ejemplo numérico presentado ilustra la flexibilidad del procedimiento para usos de diseño de ingeniería.

Ravi, Murty y Reddy (1997) aplican un heurístico mejorado titulado simulación recocida no equilibrada, NESA, para encontrar: 1) el óptimo global del costo del sistema de dos clases de sistemas complejos sujeto a restricciones de confiabilidad del sistema, y 2) el número óptimo de redundancias que maximicen la confiabilidad del sistema, sujeto a restricciones de costo del sistema, peso y volumen en un sistema mixto multiestado.

La eficiencia del NESA en la solución de los dos tipos de problemas es demostrado a partir de la comparación de sus resultados con los resultados obtenidos aplicando simulación recocida tradicional, SA. NESA resultó ser un algoritmo que converge más rápido a la solución global en comparación con SA para todos los problemas analizados. En este artículo presentan soluciones para sistemas con un máximo de 15 nodos

Tillman, Hwang y Kuo (1977) clasifican los problemas de optimización dentro de dos formatos generales:

1. Encontrar el número óptimo de redundancias, x_j, para una red compuesta de N subsistemas en serie cuya función objetivo, maximizar la confiabilidad de la red, R_s, está sujeta a restricciones de costos.

2. Encontrar el número óptimo de redundancias, x_j, que minimice los costos de la red sujetos a la restricción que la confiabilidad de la red, R_s, es igual o mayor que un nivel establecido.

Kuo y Prasad (2000) presentan en su investigación una amplia revisión de los métodos que se han desarrollado desde 1977 para la solución de varios problemas de optimización de la confiabilidad de sistemas, aplicación de estos métodos a varios tipos de problemas de diseño y un estudio de los algoritmos heurísticos, metaheurísticos, métodos exactos, asignación de redundancia, optimización multi-objetivo y asignación de componentes intercambiables en sistemas de confiabilidad. Concluyen que la mayoría del trabajo previo a la publicación de su artículo en el área ha sido dedicado al desarrollo de algoritmos heurísticos y metaheurísticos para resolver problemas de asignación óptima de redundancia. Un pequeño grupo de investigaciones han sido dirigidas a la solución exacta de estos problemas, dado que, como en otras aplicaciones, la solución exacta de los problemas de optimización de confiabilidad no son necesariamente deseables ya que las soluciones exactas son difíciles de obtener y aún cuando son alcanzables o posibles su utilidad es marginal.

1.3.3 Algoritmos Genéticos (AG)

En los años setenta, de la mano de John Holland surgió una de las líneas más prometedoras de la inteligencia artificial, la de los Algoritmos Genéticos (AG). Son llamados así porque se inspiran en la evolución biológica y su base genético-molecular.

Los AG son métodos sistemáticos para la solución de problemas de búsqueda y optimización que aplican los mismos métodos de la evolución biológica: selección basada en la población, reproducción sexual y mutación.

En un algoritmo genético, tras parametrizar el problema en una serie de variables, $(x_i,...,x_n)$ se codifican en un cromosoma. Todos los operadores utilizados por un algoritmo genético se aplicarán sobre estos cromosomas o sobre poblaciones de ellos. En el algoritmo genético va implícito el método para resolver el problema; son sólo parámetros de tal método los que están codificados, a diferencia de otros algoritmos evolutivos como la programación genética.

Las soluciones codificadas en un cromosoma compiten para ver cuál constituye la mejor solución (aunque no necesariamente la mejor de todas las soluciones posibles). El ambiente,

constituido por las otras soluciones, ejercerá una presión selectiva sobre la población, de forma que sólo los mejor adaptados (aquellos que resuelvan mejor el problema) sobrevivan o leguen su material genético a las siguientes generaciones, igual que en la evolución de las especies. La diversidad genética se introduce mediante mutaciones y reproducción sexual.

Por lo tanto, un algoritmo genético consiste en lo siguiente: hallar de qué parámetros depende el problema, codificarlos en un cromosoma y aplicar los métodos de la evolución: selección y reproducción sexual con intercambio de información y alteraciones que generan diversidad.

Coit y Smith enfocan sus investigaciones (1995, 1996) en la utilización de AG para resolver el problema de optimización de redes en serie-paralelo adicionando redundancia. En el primer estudio que publicaron (Coit y Smith, 1995) combinan los AG para determinar la mejor configuración de la red adicionando redundancia y las redes neuronales para ir evaluando la confiabilidad para cada candidato de diseño generado dado el tiempo requerido de computación para hacer los cálculos de la confiabilidad mediante los métodos convencionales para redes que ellos denominan de tamaño considerable, presentando ejemplos para redes con máximo 48 componentes organizados en 6 subsistemas en serie. Establecen como función objetivo minimizar costos y su única restricción es un valor mínimo de confiabilidad del sistema. En su segunda investigación (Coit y Smith, 1996) se enfocan en dar solución al problema de seleccionar los componentes y niveles de redundancia que optimice la función objetivo dadas unas restricciones de confiabilidad, costos y/o peso. La formulación del problema es maximizar la confiabilidad (Problema 1) o minimizar los costos (Problema 2) dadas unas restricciones. Utilizan un algoritmo genético para resolver cada problema y lo describen como una técnica de búsqueda evolucionada de optimización con pocas restricciones en función del tipo o tamaño del problema de diseño. En la formulación del problema hay un número específico de subsistemas (conjunto de componentes, arcos y nodos, de la red) y, para cada subsistema, hay múltiples componentes que pueden ser seleccionados, con posibilidades de reemplazo, y usados en paralelo. Los autores enfatizan que los AG han demostrado convergir a la solución óptima para muchos problemas de diversidad y gran dificultad, sin embargo la solución óptima global no puede ser garantizada. La habilidad de los AG para encontrar una solución eficientemente depende de las propiedades que el usuario le suministre en términos de codificación, operadores de reproducción y medidas de aptitud (“fitness”). En este artículo, Coit y Smith hacen un breve resumen de la literatura disponible sobre la solución de problemas de confiabilidad utilizando

AG. Comentan que los AG han sido utilizados para resolver muchos problemas de dificultad en la ingeniería y son particularmente efectivos para problemas de optimización combinatoria de gran complejidad y tamaño. Algunos estudios han usado AG para encontrar soluciones que maximicen la confiabilidad y satisfagan restricciones específicas de costos. El heurístico ha sido comparado con soluciones encontradas por programación no lineal y entera generando aproximaciones cercanas al óptimo usando menos recursos y tiempo computacional lo cual se ve reflejado en ahorros en los costos de estudio y flexibilidad en el análisis de alternativas de parámetros, entre otros.

Berna, Freya y Smith (1997) presentan un algoritmo genético con codificación, inicialización y operadores locales de búsqueda especializados para optimizar el diseño de topologías de redes de comunicación.

La principal complicación que estos autores enuncian es la necesidad de calcular la confiabilidad de la red completamente conectada, probabilidad de que todos los pares de nodos se encuentren comunicados, para poder estimar la función de aptitud asociada a cada posible solución dentro de la población. Este cálculo hace que el problema sea uno de tipo pesado en términos computacionales. Para la estimación de la confiabilidad de las redes proponen la aproximación por Simulación de Monte Carlo.

La efectividad del algoritmo propuesto es demostrada presentando los resultados para 79 problemas generados aleatoriamente que tienen configuraciones de redes con máximo 20 nodos y 55 arcos.

Kuo y Prasad (2000) presentan algunas de las ventajas de los AG expuestas por diversos autores que han trabajado desde 1977 el problema de optimización de la confiabilidad de sistemas. Los expertos han enunciado que los AG son muy útiles para resolver problemas de optimización discreta complejos y no requieren tratamientos matemáticos sofisticados. Además, pueden ser fácilmente diseñados e implantados en un computador. Una ventaja importante de AG es que encuentra buenas soluciones muchas veces la óptima o cercana a la óptima. Las soluciones alcanzadas por AG proveen gran flexibilidad para la toma de decisiones de diseños de confiabilidad. Sin embargo, hay algunas dificultades del heurístico en la determinación de los valores adecuados para los parámetros y la función de penalidad por infactibilidad de las soluciones.

1.4 Organización

Después de esta introducción, una descripción detallada del problema y un ejemplo que ilustra su complejidad son presentados en el Capítulo 2. El tercer capítulo describe la metodología y conceptos adicionales. El Capítulo 4 contiene el análisis de los resultados. El Capítulo 5 presenta las conclusiones finales y lineamientos para el trabajo futuro.

Adicional a los capítulos, este trabajo tiene dos apéndices. Apéndice A contiene los programas desarrollados en Matlab 7.0.1. y el Apéndice B lista los resultados obtenidos para los experimentos factoriales realizados.

2 Descripción del Problema

La confiabilidad de la red es la probabilidad de que una red funcione, exista conexión entre el nodo origen y nodo destino, durante un período de tiempo establecido. Considerando esta definición de confiabilidad se decidió asignar confiabilidades bajas a los arcos de la red, menores de 0.5, con el fin de optimizar una red que se ajuste a modelos reales donde el tiempo esperado de funcionamiento de la red es suficientemente largo.

Este capítulo describe en detalle el problema de optimización de redes complejas con confiabilidades bajas asociadas a sus arcos mediante la ubicación de arcos redundantes. Inicialmente se expone la complejidad del problema en términos del planteamiento del modelo de optimización y su solución computacional. En la segunda parte se presenta un ejemplo de una red compleja pequeña (5 nodos) para ilustrar el problema, explicar algunos conceptos importantes y exponer parte de la notación necesaria para los siguientes capítulos.

2.1 Complejidad del problema

La optimización de redes con probabilidad de falla mediante la adición de arcos, algunos posiblemente en redundancia, es un problema de programación no lineal entera cuya complejidad radica en la evaluación de la confiabilidad de las redes por ser de tipo complejo ya que todos sus nodos no se encuentran conectados ni en serie ni en paralelo. El cálculo de la función objetivo del problema por ser una expresión matemática extensa y compleja exige gran cantidad de memoria computacional que crece a medida que las redes se hacen grandes. Para el problema de optimización planteado deben considerarse todas las posibles conexiones (arcos) entre las combinaciones de pares de nodos, a lo que en la literatura se le conoce como redes completamente conectadas. El problema de optimización de redes mediante arcos redundantes ha sido resuelto por diversos investigadores y publicado en la literatura presentando resultados para redes complejas con máximo 20 nodos y 55 arcos (Dengiz, Antiparmak y Smith (1997)).

Este trabajo de tesis plantea resolver el problema de optimización mediante dos algoritmos aproximados de optimización, un algoritmo genético (AG) y un algoritmo desarrollado y propuesto por la autora basado en programación lineal entera secuencial (PLES), el cual se proyecta sea más eficiente que el AG en términos de tiempo y aproximación, evaluando la

función objetivo exactamente para redes con máximo de 7 nodos y por Simulación de Monte Carlo para redes de entre 8 y 20 nodos.

Para calcular la confiabilidad exacta de la red se utiliza su función de estructura. La función de estructura de la red, $\Phi_R(\mathbf{x})$, es igual a 0 si la red falla y 1 si la red funciona. Con el fin de calcular $\Phi_R(\mathbf{x})$ se ha determinado trabajar con el método de trazado de rutas que consiste en determinar todas las rutas mínimas que hay entre los nodos inicio y final a partir de la matriz de conexión de la red, una matriz triangular superior que relaciona los nodos. Si el arco k conecta los nodos i y j, las posiciones (i,j) y (j,i) de la matriz de conexión serán x_k (x_k es igual a 0 si el arco k falla y 1 si el arco k funciona). Una ruta mínima es el conjunto mínimo de componentes que aseguran la operación y funcionamiento de la red. Para determinar las rutas mínimas de la red se aplica el algoritmo de trazado de rutas propuesto por Fotuhi, Billinton, Munian y Vinayagam (2004). Este método parte de la matriz de conexión de la red y consta de dos pasos, primero, trazado de todas las rutas mínimas y su almacenamiento en un formato específico, y segundo, determinación de todas las rutas mínimas. Después de determinar las rutas mínimas, se construye una nueva red paralelo-serie ubicando los componentes de las rutas mínimas en serie y éstas a su vez en paralelo para calcular la función de estructura de la red como se muestra en la ecuación (2.2).

Una vez calculada la función de estructura de la red se puede calcular la confiabilidad de la red, R_R, la cual es definida en la Ecuación (2.1) como:

$$R_R = \Pr\{\Phi_R(\mathbf{x}) = 1\} = E[\Phi_R(\mathbf{x})] \tag{2.1}$$

Para poder resolver el problema de optimización, es necesario asumir que la red a estudiar cumple lo siguiente (Coit y Smith, 1996):

1. Las confiabilidades de los componentes (arcos) son conocidas y determinísticas.
2. Las probabilidades de falla de cada arco individual son independientes.
3. Toda redundancia es redundancia activa sin reparar. La redundancia activa sin reparar indica que la red continúa operando aunque uno de sus arcos falle y esta falla no es reparada hasta que toda la red falle.

Este problema no lineal entero tiene como función objetivo maximizar la confiabilidad de la red sujeta a una restricción de presupuesto de inversión.

Notación:

$R_{RT}(\mathbf{yo}, \mathbf{y})$: confiabilidad total de la red con arcos adicionales

a: número de posibles combinaciones de pares de nodos $\left(a = n(n\text{-}1)/2\right)$ *

* El número de posibles combinaciones de pares de nodos, a, va a denominarse en este trabajo como número de tipos de arcos, es decir, cada posible conexión (combinación) de dos nodos va a ser considerada como un tipo de arco.

C_i: costo de adicionar a la red un arco tipo i, i=1, 2,..., a

P: presupuesto total disponible

$maxred_i$: número máximo de arcos tipo i en la red, i=1, 2,..., a

yo: vector de estado de la red original (vector binario de tamaño a donde 1 significa que el arco existe en la red original)

n: número de nodos de la red

Variables de Decisión:

y_i: Número de arcos i adicionales asignados a la red (posiblemente en redundancia)

$y_i = 0, 1, 2, ..., maxred_i - yo_i$; i = 1, 2, ..., a

Problema :

$$\text{Maximizar } R_{RT}(\mathbf{yo}, \mathbf{y}) = E(\phi(\mathbf{yo}, \mathbf{y}))$$

$$\text{Sujeto a } \sum_{i=1}^{a} C_i y_i \leq P$$

El número máximo de arcos en redundancia por cada tipo de arco de la red, *maxred*, es un valor que se establece en función de las especificaciones de la red a trabajar y en caso de no existir esta restricción en el diseño de la red se puede indicar un valor grande para la variable *maxred*.

Las confiabilidades y costos asociados a los arcos de la red es información que se estima previamente determinada y se presume conocida. No se plantea relación directa entre las confiabilidades y los costos de los arcos redundantes dado que en casos de sistemas reales como una red de electricidad, una red de computadoras o una red de transporte, la adición de una conexión (arco) de alto costo no implica una confiabilidad alta de la conexión, de igual manera una conexión de bajo costo no esta asociada necesariamente a una confiabilidad baja. Un ejemplo de esto sería el estimar la adición de una conexión de una red eléctrica cuyo costo es alto por estar instalada en un terreno montañoso y su confiabilidad es baja por estar expuesta a fuertes vientos.

2.2 Ejemplo

La solución del problema de optimización para una red compleja pequeña es ilustrado con el desarrollo del siguiente ejemplo cuyo diagrama de red se muestra en la Figura 6.

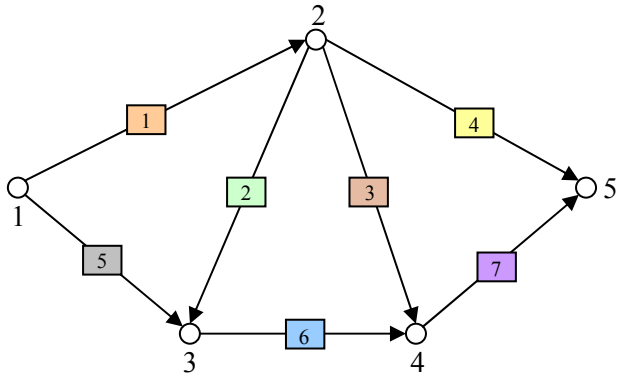


Figura 6. Red compleja

La red tiene 7 arcos y 5 nodos. Cada arco tiene una confiabilidad asociada la cual es denotada por r_j (j=1,2,…,7) cuyos valores para el ejemplo son presentados en la Tabla 1. La red funciona si existe conexión entre los nodos origen y destino, 1 y 5 en el ejemplo.

Tabla 1. Confiabilidades de los arcos

Nodos	**1**	**2**	**3**	**4**	**5**
1		22.69%	25.88%		
2			27.81%	31.32%	22.77%
3				30.70%	
4					38.17%
5					

Empleando la Ecuación (2.2), deducida en la sección de metodología se calcula el valor de la función de estructura, $\Phi_{RO}(\mathbf{yo})$, para cada uno de los 2^7 posibles arreglos del vector de estado $\mathbf{yo} = (yo_1, yo_2, \ldots, yo_7)$ de la red ejemplo.

$$\Phi_{RO}(\mathbf{yo}) = 1 - [(1 - yo_1 yo_4)(1 - yo_1 yo_3 yo_7)(1 - yo_5 yo_6 yo_7)(1 - yo_1 yo_2 yo_6 yo_7)] \qquad (2.2)$$

El valor de la confiabilidad de la red original se calcula reemplazando las variables de estado asociadas a cada arco, yo_i, por las confiabilidades de los arcos originales, ro_i, en la expresión de la función de estructura expandida como se presenta en la Ecuación (2.3):

$$\begin{aligned} R_{RO} &= ro_1ro_4 + ro_1ro_3ro_7 + ro_5ro_6ro_7 + ro_1ro_2ro_6ro_7 - ro_1ro_3ro_4ro_7 - ro_1ro_2ro_5ro_6ro_7 \\ &\quad - ro_1ro_2ro_3ro_6ro_7 - ro_1ro_3ro_5ro_6ro_7 - ro_1ro_2ro_4ro_6ro_7 - ro_1ro_4ro_5ro_6ro_7 \\ &\quad + ro_1ro_2ro_3ro_5ro_6ro_7 + ro_1ro_2ro_4ro_5ro_6ro_7 + ro_1ro_2ro_3ro_4ro_6ro_7 + ro_1ro_3ro_4ro_5ro_6ro_7 \\ &\quad - ro_1ro_2ro_3ro_4ro_5ro_6ro_7 \end{aligned} \tag{2.3}$$

$$R_{RO} = 10.26\%$$

La confiabilidad de 10.26% calculada para la red original puede mejorarse añadiendo arcos que pueden corresponder a arcos paralelos redundantes a los ya existentes o arcos que conecten nodos que no se encontraban comunicados directamente. En las Tablas 2 y 3 se presentan las confiabilidades y los costos unitarios de los diez tipos de arcos que pueden adicionarse a la red. Se tiene una restricción de un presupuesto de inversión de $99,000.

Tabla 2. Confiabilidades de arcos redundantes

Nodos	1	2	3	4	5
1		39.23%	38.74%	33.30%	31.00%
2			33.34%	38.83%	39.32%
3				38.24%	38.08%
4					32.28%
5					

Tabla 3. Costos de arcos redundantes

Nodos	1	2	3	4	5
1		$ 42,639	$ 15,944	$ 565	$ 314
2			$ 571	$ 18,536	$ 54,505
3				$ 8,167	$ 6,904
4					$ 425
5					

Empleando la Ecuación (2.4) se calcula el valor de la función de estructura de la red de arcos redundantes, $\Phi_{RR}(\mathbf{y})$, para cada uno de los 2^{10} posibles arreglos del vector de estado $\mathbf{y} = (y_1, y_2, \ldots, y_{10})$ de la red de la Figura 7.

$$\Phi_{RR}(\mathbf{y}) = 1 - [(1-y_9)(1-y_1y_4)(1-y_5y_{10})(1-y_7y_8)(1-y_5y_6y_7)(1-y_1y_2y_{10})(1-y_1y_3y_7)(1-y_1y_2y_6y_7)] \quad (2.4)$$

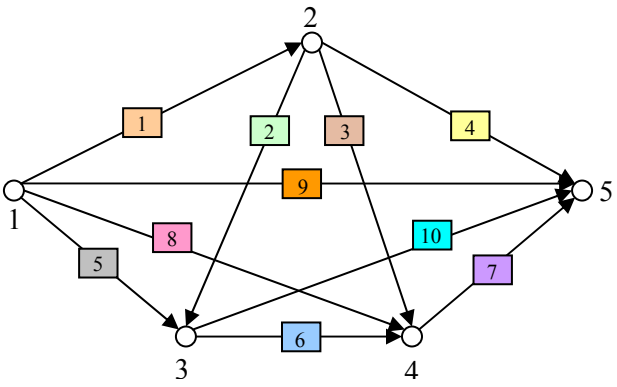


Figura 7. Red compleja con todos los posibles arcos redundantes

Para calcular la confiabilidad total de cada arco i, r_{Ti}, considerando la posibilidad que se encuentre o no en redundancia se utiliza la Ecuación (2.5):

$$r_{Ti} = 1 - (1 - r_{Oi})(1 - r_{Ri})^{y_i} \quad (2.5)$$

r_{Oi}: confiabilidad del arco i en la red original

r_{Ri}: confiabilidad del arco i cuando es redundante

Con el uso de la herramienta de optimización de Excel se resuelve el problema del ejemplo y se logra maximizar la confiabilidad total de la red tras la aplicación de redundancia, estableciendo que el número máximo de arcos en redundancia para cada tipo de arco de la red ejemplo es dos (*maxred* = 2).

A continuación se describe brevemente el procedimiento seguido para el planteamiento y solución del problema utilizando Excel:

1. Se construye una tabla con a (10) columnas correspondientes a cada tipo de arco de la red y 2^a (2^{10}=1024) filas correspondientes a los posibles valores para el vector de estado de la red.
2. Para cada vector de estado de la red (fila) se calcula el valor de su función de estructura ($\varphi(\mathbf{y})$) y el resultado se ubica en la columna a+1 (11) como se presenta en la Tabla 4.

Tabla 4. Cálculo de la función de estructura ($\varphi(\mathbf{y})$) para cada posible vector de estado de la red

y_1	y_2	y_3	y_4	y_5	y_6	y_7	y_8	y_9	y_{10}	**Φ(y)**
1	1	1	1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1	1	0	1
1	1	1	1	1	1	1	1	0	1	1
1	1	1	1	1	1	1	1	0	0	1
:	:	:	:	:	:	:	:	:	:	:
0	0	0	0	0	0	0	1	0	0	0
0	0	0	0	0	0	0	0	1	0	1
0	0	0	0	0	0	0	0	0	1	0
0	0	0	0	0	0	0	0	0	0	0

3. Se construye una nueva tabla con a (10) columnas correspondientes a la confiabilidad total de cada tipo de arco de la red (r_T) y 2^a (2^{10}=1024) filas con los valores de confiabilidad de los arcos para los vectores de estado correspondientes, ubicando en las posiciones donde la variable de estado es 1 (indicando que el arco existe) el valor de la confiabilidad del arco (r_T)y en las posiciones donde la variable de estado es 0 (indicando que el arco no existe) el valor de la probabilidad de falla del arco (1- r_{Ti}). La fila que contiene los valores de las confiabilidades totales de cada arco es una fila que depende del vector de las variables de solución (**y**), ya que el valor de r_T depende del número de arcos en redundancia que se adicionen por cada tipo de arco como se presenta en la Ecuación (2.5).
4. Para cada vector de confiabilidades (fila), asociado a cada posible vector de estado de la red, se calcula el valor de la confiabilidad total con redundancia (R_R(y)) que es igual al producto de todos los componentes del vector (columnas) y el resultado se ubica en la columna a+1 (11) como se presenta en la Tabla 5.

Tabla 5. Cálculo de la confiabilidad total asociada a cada posible vector de estado de la red

r_{T1}	r_{T2}	r_{T3}	r_{T4}	r_{T5}	r_{T6}	r_{T7}	r_{T8}	r_{T9}	r_{T10}	R_R(y)
0.5302	**0.5621**	**0.3132**	**0.2277**	**0.5459**	**0.5720**	**0.5813**	**0.5551**	**0.5239**	**0.6166**	
0.5302	0.5621	0.3132	0.2277	0.5459	0.5720	0.5813	0.5551	0.5239	0.6166	0.0007
0.5302	0.5621	0.3132	0.2277	0.5459	0.5720	0.5813	0.5551	0.5239	0.3834	0.0004
0.5302	0.5621	0.3132	0.2277	0.5459	0.5720	0.5813	0.5551	0.4761	0.6166	0.0006
0.5302	0.5621	0.3132	0.2277	0.5459	0.5720	0.5813	0.5551	0.4761	0.3834	0.0004
:	:	:	:	:	:	:	:	:	:	:
0.4698	0.4379	0.6868	0.7723	0.4541	0.4280	0.4187	0.4449	0.5239	0.6166	0.0013
0.4698	0.4379	0.6868	0.7723	0.4541	0.4280	0.4187	0.4449	0.5239	0.3834	0.0008
0.4698	0.4379	0.6868	0.7723	0.4541	0.4280	0.4187	0.4449	0.4761	0.6166	0.0012
0.4698	0.4379	0.6868	0.7723	0.4541	0.4280	0.4187	0.4449	0.4761	0.3834	0.0007

5. Se construye una tabla con los valores dados para las confiabilidades de los arcos originales (r_O) y redundantes (r_R) de la red (Tabla 6).

Tabla 6. Confiabilidades de los arcos originales y redundantes de la red

	a_1	a_2	a_3	a_4	a_5	a_6	A_7	a_8	a_9	a_{10}
r_O	22.69%	27.81%	31.32%	22.77%	25.88%	30.70%	38.17%	0.00%	0.00%	0.00%
r_R	39.23%	39.34%	38.83%	39.32%	38.74%	38.24%	32.28%	33.30%	31.00%	38.08%

6. Se construye una tabla que contenga una fila con las *a* variables de respuesta (**y**) correspondiente al vector de arcos redundantes que se van a adicionar a la red. Esta tabla también debe contener una fila que corresponda a los valores de confiabilidades totales (r_T) para cada tipo de arco que se van a calcular en función de **y** y una fila de costos por cada tipo de arco que va a depender del valor de **y** (Tabla 7).

   En una celda de la hoja de Excel se calcula el valor de la confiabilidad total de la red con redundancia (R_{RT}) que es equivalente al producto suma de las columnas correspondientes a φ(**y**) y R_R (**y**).
7. En la herramienta de optimización de Excel se define la fila **y** como las celdas cambiantes (variables de respuesta), se indica que es un problema de maximización de la celda donde se haya ubicado el valor de la confiabilidad total de la red (R_{RT}), se

definen las restricciones de máxima redundancia y no negatividad para la fila **y** y se incluye la restricción de presupuesto.

Los resultados obtenidos con la herramienta de optimización de Excel se resumen en la Tabla 7 y el diagrama de red solución con los arcos adicionales se presenta en la Figura 8.

Tabla 7. Confiabilidad total de cada arco y arcos redundantes adicionales

	a_1	a_2	a_3	a_4	a_5	a_6	a_7	a_8	a_9	a_{10}
r_T	53.02%	56.21%	31.32%	22.77%	54.59%	57.20%	58.13%	55.51%	52.39%	61.66%
y	1	1	0	0	1	1	1	2	2	2
$yo + y$	2	2	1	1	2	2	2	2	2	2

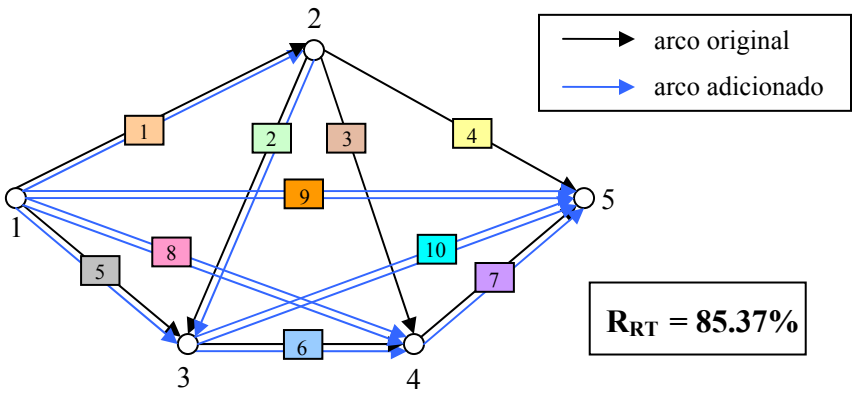


Figura 8. Red compleja con los 11 arcos redundantes

La confiabilidad de la red original era 10.26% y con la adición de 11 arcos, 8 de ellos en redundancia, se aumenta en un 75.11% alcanzando el 85.37% de confiabilidad y con una inversión de $83,312 que se encuentra dentro del presupuesto indicado.

3 Metodología

En esta investigación, redes complejas con confiabilidades bajas asociadas a sus arcos serán construidas y optimizadas mediante la adición de arcos, algunos de ellos redundantes. Este capítulo discute la metodología y conceptos requeridos para la explicación del procedimiento empleado.

La primera sección inicia con la explicación del método exacto para calcular la confiabilidad de una red compleja a partir de la función de estructura de la red. Continúa con la descripción de la metodología de trazado de rutas para determinar las rutas mínimas de la red. Finaliza con la explicación de la metodología de Simulación de Monte Carlo considerada para calcular la confiabilidad aproximada de redes complejas con más de 7 nodos. La segunda parte expone la descripción de los dos algoritmos de optimización propuestos para determinar el número y la ubicación óptima de los arcos redundantes de la red. Para la ilustración de la metodología propuesta en la investigación se utiliza el ejemplo presentado por Kuo y Zuo (2003) en la sección 5.3 del libro "Optimal Reliability Modeling" (Figura 9).

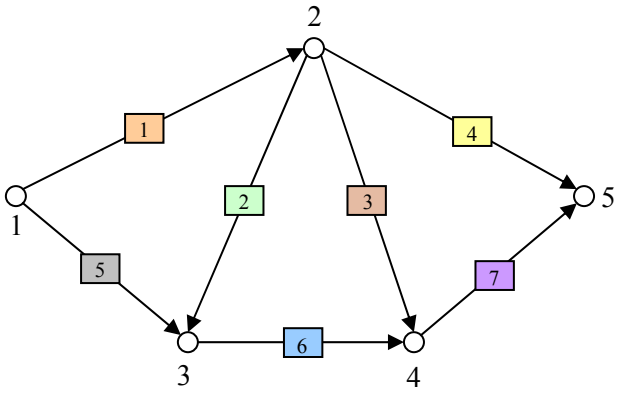


Figura 9. Red compleja

El diagrama de la Figura 10 presenta de manera general el bosquejo de la metodología con los pasos seguidos y el orden de las diferentes secciones dentro del proceso para solucionar el problema de optimización.

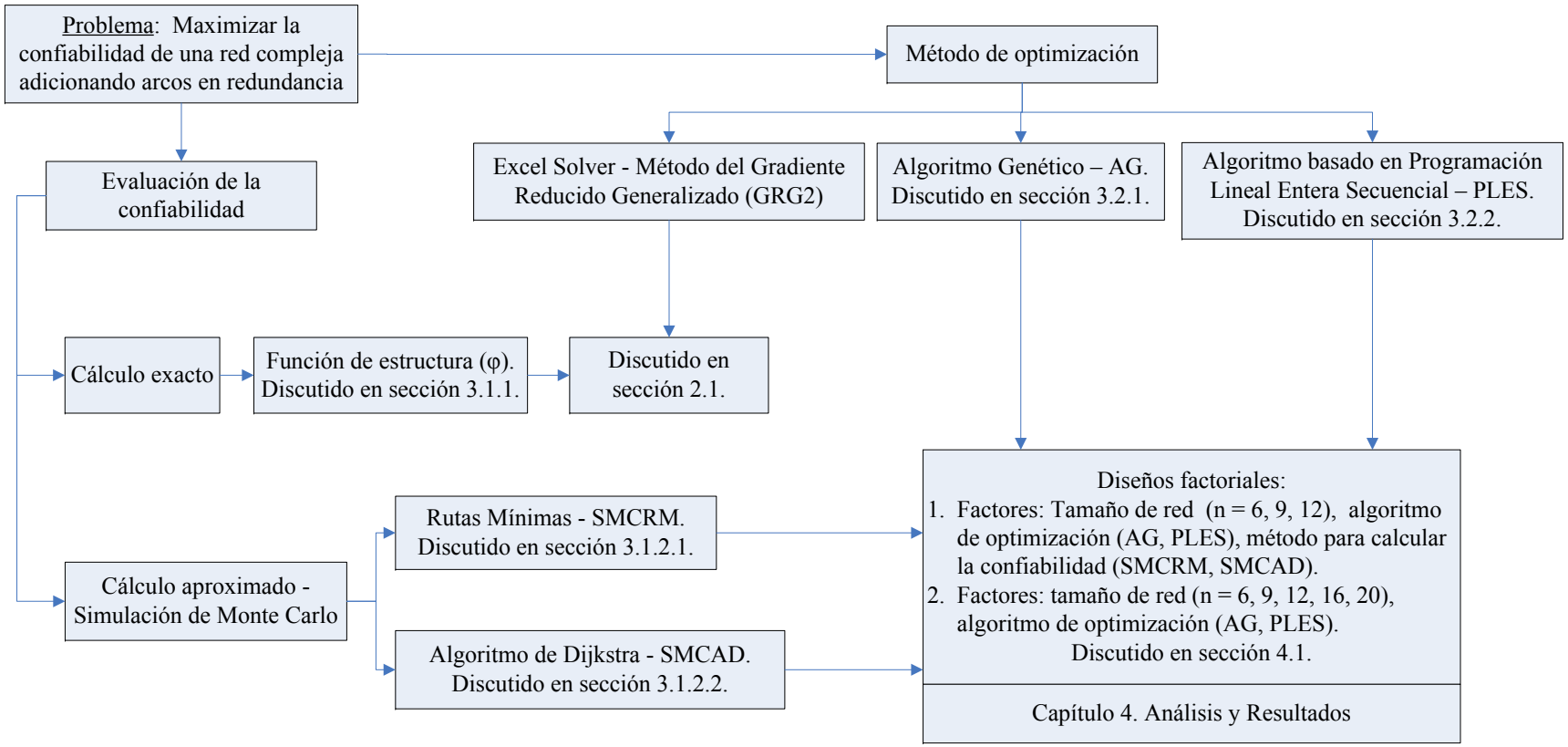


Figura 10. Diagrama general de la metodología

3.1 Cálculo de la Confiabilidad de una red

Una de las alternativas más utilizadas para calcular la confiabilidad de una red compleja es utilizando su función de estructura (Ebeling, 2005). Para puntualizar la función de estructura de la red, definimos:

$$x_i = \begin{cases} 1 & \text{si el arco i funciona} \\ 0 & \text{si el arco i falla} \end{cases}, \qquad i = 1, 2, ..., a$$

El estado de los a tipos de arcos de la red pueden organizarse en un vector de estado $\mathbf{x} = (x_1, x_2, \ldots, x_a)$. Por lo tanto, hay 2^a posibles valores para el vector de estado.

La función de estructura de la red, Φ_R, es

$\Phi_R(\mathbf{x}) = 0$ si hay al menos una conexión entre el nodo 1 y final cuando el vector de estado es $\mathbf{x}$, y $\Phi_R(\mathbf{x}) = 1$ si no hay una conexión entre el nodo 1 y final cuando el vector de estado es $\mathbf{x}$.

Existen tres métodos estándar para calcular la función de estructura de una red. Estos son:

1. Enumeración
2. Trazado de rutas
3. Condicionamiento en un elemento clave.

La confiabilidad de la red es el valor esperado de la función de estructura, es decir, la probabilidad de que la red funcione.

3.1.1 Rutas mínimas

Se ha decidido trabajar con la metodología de rutas mínimas para calcular la función de estructura de la red. Una ruta mínima es una ruta para la cual la falla de uno de sus arcos genera una falla de la ruta, es decir, el hecho que uno de los arcos de la ruta no funcione implica que no haya conexión entre los nodos origen y final de la red empleando esa ruta.

En la red ejemplo presentada en la Figura 9, los arcos han sido numerados de 1 a 7 y los nodos de 1 a 5. El objetivo es encontrar todas las rutas mínimas que hay entre los nodos 1 y 5. Para ello se construye la matriz de conexión de la red1 que indica la conexión entre los nodos. Si

no hay conexión directa entre los nodos i y j, la posición (i,j) de la matriz de conexión será cero. Dado que un nodo siempre está conectado con él mismo, los valores de la diagonal de la matriz son uno. Si el arco k conecta los nodos i y j, la posición (i,j) de la matriz de conexión será x_k.

La matriz de conexión es una matriz triangular superior, dado que la red es unidireccional, que será el punto de partida para determinar el número y configuración de las rutas mínimas que tiene la red.

Matriz de conexión de la red (nodo a nodo)

$$C = \begin{bmatrix} 1 & x_1 & x_5 & 0 & 0 \\ 0 & 1 & x_2 & x_3 & x_4 \\ 0 & 0 & 1 & x_6 & 0 \\ 0 & 0 & 0 & 1 & x_7 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

Existen diversos métodos propuestos en la literatura para determinar las rutas mínimas de una red, dos de los más conocidos son el método de remoción de nodos y el método de multiplicación de la matriz. En el método de remoción todos los nodos que no son ni el inicio ni el final de la red son removidos a través de una reducción secuencial de la matriz de conexión hasta que ésta queda de tamaño 2 x 2. Como resultado se obtienen las rutas que existen solamente entre los dos nodos resultantes de interés, inicio y final de la red (Kuo y Zuo, 2003). En el método de multiplicación la matriz de conexión es multiplicada por ella misma un número de veces hasta que la matriz resultante permanece incambiable. El resultado final presenta todas las rutas entre todos los pares de nodos de la red (Billinton y Allan, 1994). Estos dos métodos emplean la ley fundamental y la ley de idempotencia del sistema lógico denominado álgebra booleana las cuales enuncian que la suma de un elemento con el número uno da por resultado el número uno y que la suma y multiplicación de dos elementos iguales da por resultado el mismo elemento (ej. $x_1+1 = 1$, $x_1.x_1 = x_1$, $x_1+x_1 = x_1$).

Para la generación de las rutas mínimas de la red se emplea el algoritmo de trazado de rutas propuesto por Fotuhi, Billinton, Munian y Vinayagam (2004). Ésta técnica es aplicable a redes complejas, y considera arcos tanto unidireccionales como bidireccionales. Similar a las otras dos técnicas explicadas para la generación de rutas mínimas de una red, el primer paso de este algoritmo es determinar la matriz de conexión de la red (matriz C).

El proceso del algoritmo consta de dos pasos:

1. Trazado de todas las rutas mínimas y su almacenamiento en un formato específico (ver diagrama de flujo Figura 11).
2. Determinación de todas las rutas mínimas (ver diagrama de flujo Figura 12).

La terminología asociada al algoritmo se lista a continuación:

- Nodo inicio, nodo salida: nodos de interés.
- Número de rama: número asignado a cada componente por orden de ingreso en la tabla de almacenamiento.
- Rama padre: número de rama del elemento desde el cual el componente actual ha sido ramificado.
- Número de nodo: número de la columna de un elemento.
- Estatus: bandera que indica si un componente ha sido ramificado y es representada por 1 si el elemento puede ser ramificado más adelante, y 0 si el elemento no puede ser ramificado.
- Nodo previo: número de rama del elemento.
- Nombre del componente: nombre del elemento (arco) conectado entre dos nodos (número del nodo y el nodo previo).
- Fila actual: número del nodo del elemento.
- Contador: número de veces en que debe repetirse el proceso hasta trazar todas las rutas mínimas.
- Nodo padre: elemento en la rama padre.

El algoritmo es ilustrado usando el diagrama de la red unidireccional presentada en la Figura 9 y su matriz de conexión C.

Inicialización de variables:

Las variables temporales primero son inicializadas con los siguientes valores:

- Nodo previo = 0
- Contador = 1
- Fila actual = nodo inicio = 1
- Rama padre = 0

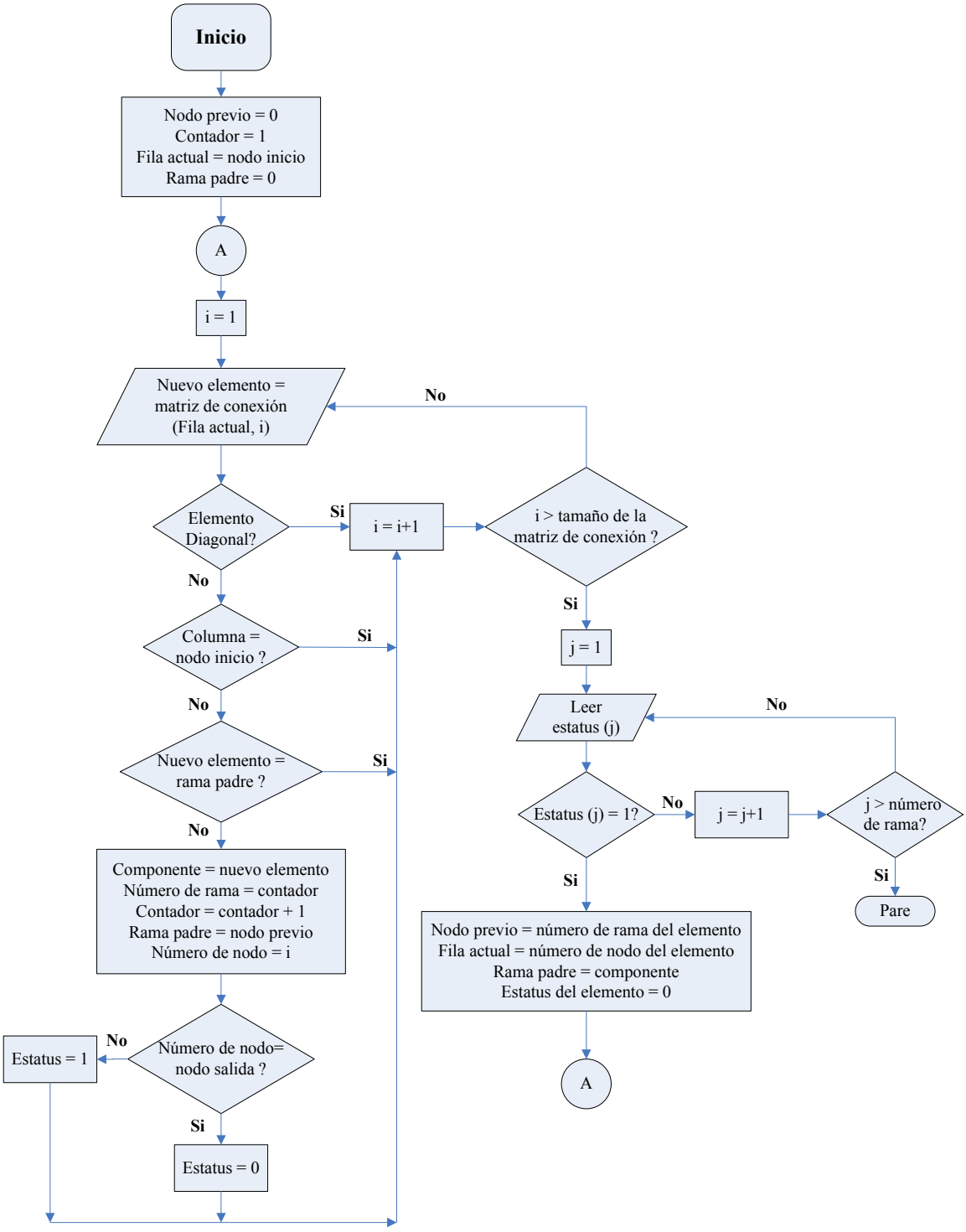


Figura 11. Diagrama de flujo del almacenamiento de las rutas

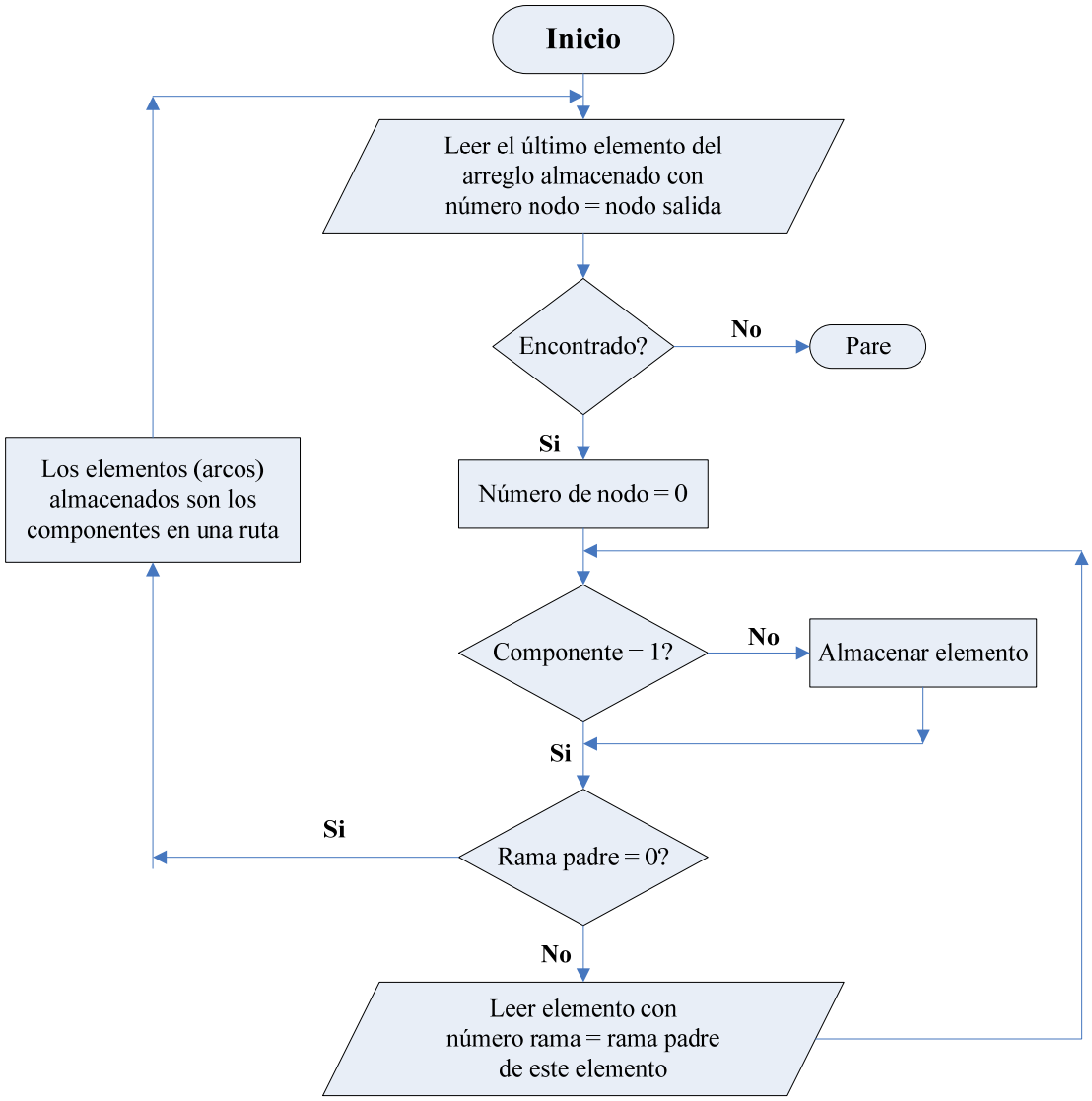


Figura 12. Diagrama de flujo para determinación de las rutas

Paso 1 – Almacenamiento de rutas: siguiendo el diagrama de flujo presentado en la Figura 11 se obtienen las tablas que se muestran a continuación:

Tabla 8. Iniciación de trazado y almacenamiento de rutas

Nodo Número	2	3
Rama Padre	0	0
Número Rama	1	2
Componente	1	5
Estatus	1	1

Tabla 9. Continuación trazado y almacenamiento de rutas

Nodo Número	2	3	3	4	5
Rama Padre	0	0	1	1	1
Número Rama	1	2	3	4	5
Componente	1	5	2	3	4
Estatus	0	1	1	1	0

Nodo previo = 1; Contador = 3; Fila actual = 2; Rama padre = 1

Tabla 10. Continuación trazado y almacenamiento de rutas

Nodo Número	2	3	3	4	5	4
Rama Padre	0	0	1	1	1	2
Número Rama	1	2	3	4	5	6
Componente	1	5	2	3	4	6
Estatus	0	0	1	1	0	1

Nodo previo = 2; Contador = 6; Fila actual = 3; Rama padre = 5

Tabla 11. Continuación trazado y almacenamiento de rutas

Nodo Número	2	3	3	4	5	4	4
Rama Padre	0	0	1	1	1	2	3
Número Rama	1	2	3	4	5	6	7
Componente	1	5	2	3	4	6	6
Estatus	0	0	0	1	0	1	1

Nodo previo = 3; Contador = 7; Fila actual = 3; Rama padre = 2

Tabla 12. Continuación trazado y almacenamiento de rutas

Nodo Número	2	3	3	4	5	4	4	5
Rama Padre	0	0	1	1	1	2	3	4
Número Rama	1	2	3	4	5	6	7	8
Componente	1	5	2	3	4	6	6	7
Estatus	0	0	0	0	0	1	1	0

Nodo previo = 4; Contador = 8; Fila actual = 4; Rama padre = 3

Tabla 13. Continuación trazado y almacenamiento de rutas

Nodo Número	2	3	3	4	5	4	4	5	5
Rama Padre	0	0	1	1	1	2	3	4	6
Número Rama	1	2	3	4	5	6	7	8	9
Componente	1	5	2	3	4	6	6	7	7
Estatus	0	0	0	0	0	0	1	0	0

Nodo previo = 6; Contador = 9; Fila actual = 4; Rama padre = 6

Tabla 14. Trazado y almacenamiento de rutas completo

Nodo Número	2	3	3	4	5	4	4	5	5	5
Rama Padre	0	0	1	1	1	2	3	4	6	7
Número Rama	1	2	3	4	5	6	7	8	9	10
Componente	1	5	2	3	4	6	6	7	7	7
Estatus	0	0	0	0	0	0	0	0	0	0

Nodo previo = 7; Contador = 10; Fila actual = 4; Rama padre = 6

Paso 2 – Determinación de rutas: siguiendo el diagrama de flujo presentado en la Figura 12 se determinan las rutas mínimas de la red como se muestra en las siguientes tablas:

Tabla 15. Determinación de la primera ruta

Nodo Número	2	3	3	4	5	4	4	5	5	5
Rama Padre	0	0	1	1	1	2	3	4	6	7
Número Rama	1	2	3	4	5	6	7	8	9	10
Componente	1	5	2	3	4	6	6	7	7	7
Estatus	0	0	0	0	0	0	0	0	0	0

Tabla 16. Determinación de la segunda ruta

Nodo Número	2	3	3	4	5	4	4	5	5	5
Rama Padre	0	0	1	1	1	2	3	4	6	7
Número Rama	1	2	3	4	5	6	7	8	9	10
Componente	1	5	2	3	4	6	6	7	7	7
Estatus	0	0	0	0	0	0	0	0	0	0

Tabla 17. Determinación de la tercera ruta

Nodo Número	2	3	3	4	5	4	4	5	5	5
Rama Padre	0	0	1	1	1	2	3	4	6	7
Número Rama	1	2	3	4	5	6	7	8	9	10
Componente	1	5	2	3	4	6	6	7	7	7
Estatus	0	0	0	0	0	0	0	0	0	0

Tabla 18. Determinación de la cuarta ruta

Nodo Número	2	3	3	4	5	4	4	5	5	5
Rama Padre	0	0	1	1	1	2	3	4	6	7
Número Rama	1	2	3	4	5	6	7	8	9	10
Componente	1	5	2	3	4	6	6	7	7	7
Estatus	0	0	0	0	0	0	0	0	0	0

La red ejemplo tiene entonces cuatro rutas mínimas:

RM1 = {1, 4} RM 2 = {1, 3, 7}

RM 3 = {5, 6, 7} RM 4 = {1, 2, 6, 7}

Una vez determinadas las rutas mínimas, se construye una nueva red paralelo-serie ubicando los componentes de las rutas mínimas en serie y ésta a su vez en paralelo:

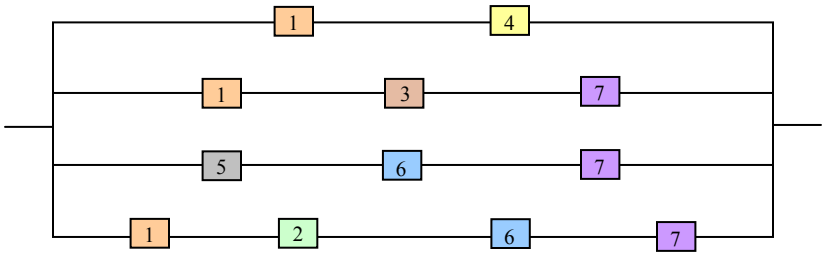


Figura 13. Red de rutas mínimas paralelo-serie (R1)

La función de estructura para un sistema con *m* componentes en serie se calcula de la siguiente manera:

$$\Phi(\mathbf{x}) = \min\{x_1, x_2, ..., x_m\} = x_1 x_2 ... x_m \tag{3.1}$$

La función de estructura para un sistema con *m* componentes en paralelo se calcula de la siguiente manera:

$$\Phi(\mathbf{x}) = \max\{x_1, x_2, ..., x_m\} = 1 - (1 - x_1)(1 - x_2)...(1 - x_m) \tag{3.2}$$

Empleando las Ecuaciones (3.1) y (3.2) y las rutas mínimas, calculamos la función de estructura de la red como:

$$\Phi_{RO}(\mathbf{x}) = 1 - [(1 - x_1 x_4)(1 - x_1 x_3 x_7)(1 - x_5 x_6 x_7)(1 - x_1 x_2 x_6 x_7)] \tag{3.3}$$

Una vez calculada la función de estructura de la red se puede calcular la confiabilidad de la red original, R_{RO}, la cual es definida como:

$$R_{RO} = \Pr\{\Phi_{RO}(\mathbf{xo}) = 1\} = E[\Phi_{RO}(\mathbf{xo})] \tag{3.4}$$

El valor de la confiabilidad de la red puede ser mejorado mediante la adición de arcos que pueden corresponder a arcos paralelos redundantes a los ya existentes o arcos que conecten

nodos que no se encontraban conectados directamente. Para la red ejemplo de cinco nodos existen 10 posibles conexiones de pares de nodos de la red ($a = n(n\text{-}1)/2 = 5(4)/2 = 10$) (Figura 14), a tipos de arcos que pueden ser adicionados a la red original considerando la restricción de número máximo de arcos entre cada par de nodos.

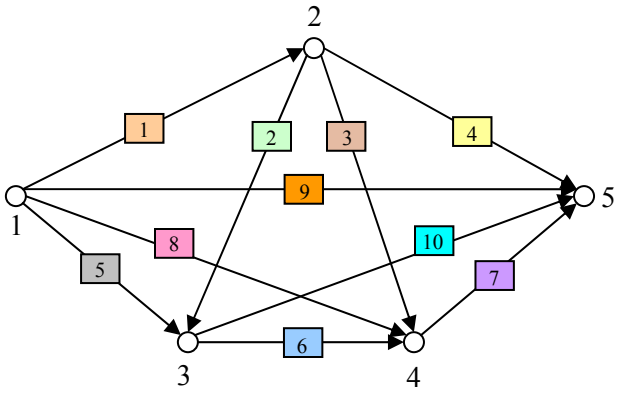


Figura 14. Red compleja con todos los posibles arcos redundantes

Se debe calcular una nueva función de estructura para la red completa con todos los posibles arcos redundantes, Φ_{RR}, con la aplicación del algoritmo de trazado de rutas.

Siguiendo el procedimiento descrito previamente para determinar todas las rutas mínimas, de la red de arcos redundantes obtenemos ocho rutas mínimas listadas a continuación:

RM1 = {9} RM 2 = {1, 4}

RM 3 = {5, 10} RM 4 = {8, 7}

RM 5 = {5, 6, 7} RM 6 = {1, 2, 10}

RM 7 = {1, 3, 7} RM 8 = {1, 2, 6, 7}

Empleando las Ecuaciones (3.1) y (3.2) y las rutas mínimas, calculamos la función de estructura de la red de arcos redundantes como:

$$\Phi_{RR}(\mathbf{x}) = 1 - [(1-x_9)(1-x_1x_4)(1-x_5x_{10})(1-x_7x_8)(1-x_5x_6x_7)(1-x_1x_2x_{10})(1-x_1x_3x_7) \\ (1-x_1x_2x_6x_7)] \qquad (3.5)$$

Cada arco redundante tiene una confiabilidad asociada, r_{Ri}, posiblemente distinta a la confiabilidad de los arcos de la red original, r_{Oi}. La confiabilidad total de cada arco i, r_{Ti},

considerando la posibilidad que se encuentre en redundancia se calcula con la aplicación de la siguiente ecuación:

$$r_{Ti} = 1 - (1 - r_{Oi})(1 - r_{Ri})^{y_i}$$
r_{Oi}: confiabilidad del arco i en la red original
r_{Ri}: confiabilidad del arco i cuando es redundante
y_i: número de arcos i adicionales asignados a la red (3.6)

El cálculo exacto de la confiabilidad para redes complejas con todos los pares de arcos conectados es computacionalmente exigente. La metodología descrita es programada en Matlab 7.0.1. empleando para la evaluación de la función de estructura de la red la herramienta de variables simbólicas (Symbolic Math Toolbox 3.1.1.) obteniendo resultados exactos de confiabilidad para redes con máximo siete nodos (a = 21 arcos).

El algoritmo de trazado de rutas programado para determinar todas las posibles rutas mínimas de una red unidireccional demanda gran cantidad de tiempo computacional y éste incrementa a medida que la red crece en número de nodos. Una red dirigida completamente conectada con n nodos, con flujo desde el nodo de menor índice hacia el nodo de mayor índice, tiene 2^{n-2} rutas mínimas. Esto implica que, por ejemplo, una red con siete nodos tiene 32 rutas mínimas demandando al programa aproximadamente 4 minutos para estimarlas, en comparación con una red de 18 nodos que tiene 65,536 rutas mínimas consumiendo para la determinación de estas rutas aproximadamente 172 horas (7.17 días) de tiempo en computadora. Adicional a la determinación de las rutas mínimas, el cálculo exacto de la confiabilidad requiere establecer la función de estructura para cada red considerando todas las rutas mínimas; una vez construida la función de estructura como se presenta en la Ecuación (3.5), la expresión debe ser expandida, por existir dependencia entre las rutas mínimas, aplicando la ley distributiva de la multiplicación y la ley de idempotencia del álgebra booleana dado que las variables de la función son binarias (dos posibles estados), proceso que toma para una red de 7 nodos aproximadamente 15 horas. Se presentaron limitaciones de memoria virtual de computadora dados los recursos disponibles para continuar con redes de mayor tamaño. Este primer paso de determinar las rutas mínimas, construir la función de estructura de la red y expandirla, se ejecuta una sola vez para cada tamaño de red y son almacenados para ser utilizados en los algoritmos de optimización. El cálculo exacto de la confiabilidad de una red en la rutina desarrollada en Matlab implica la

evaluación de la función de estructura expandida en el vector de confiabilidades de los arcos de la red y esto puede tomar aproximadamente 36 minutos para una red de 7 nodos. Los tiempos presentados fueron calculados empleando una computadora Intel Pentium M, con procesador de 1.6 GHz y 512 MB de memoria RAM.

3.1.2 Simulación Monte Carlo

3.1.2.1 Estimación de la confiabilidad empleando Rutas Mínimas

Ramírez y Coit (2005) describen una metodología de simulación de Monte Carlo, SMC, para estimar la confiabilidad de una red multiestado. Esta metodología ha sido adaptada para estimar la confiabilidad de las redes objeto de estudio. A continuación se describe brevemente el procedimiento seguido para calcular la confiabilidad de una red compleja empleando SMC (ver diagrama de flujo en la Figura 15):

Paso 1: Se genera un vector de tamaño *a* con valores aleatorios uniformes entre 0 y 1.

Paso 2: Se construye un vector de estado para la red a partir de la comparación de cada posición del vector aleatorio, correspondiente a un posible arco de la red, con la confiabilidad asociada al arco, si el valor aleatorio es menor que la confiabilidad del arco se ubica un 1 en la posición del arco dentro del vector de estado, de lo contrario se asigna un 0.

Paso 3: Las rutas mínimas de la red son traducidas a vectores binarios donde el 1 representa la participación del arco dentro de la ruta. El vector de estado es comparado contra cada ruta mínima y se contabiliza un éxito si el vector de estado es mayor o igual que alguna de las rutas mínimas.

Paso 4: Cada iteración de los pasos 1-3 es denominada corrida de simulación. Se realizan *NS* corridas de simulación y la proporción de éxitos en *NS* corridas es el valor estimado de la confiabilidad.

SMC es un método no determinístico usado para aproximar numéricamente expresiones matemáticas complejas y difíciles de evaluar con exactitud. El método de Monte Carlo proporciona soluciones aproximadas a una gran variedad de problemas matemáticos posibilitando la realización de experimentos con muestreos estadísticos en una computadora.

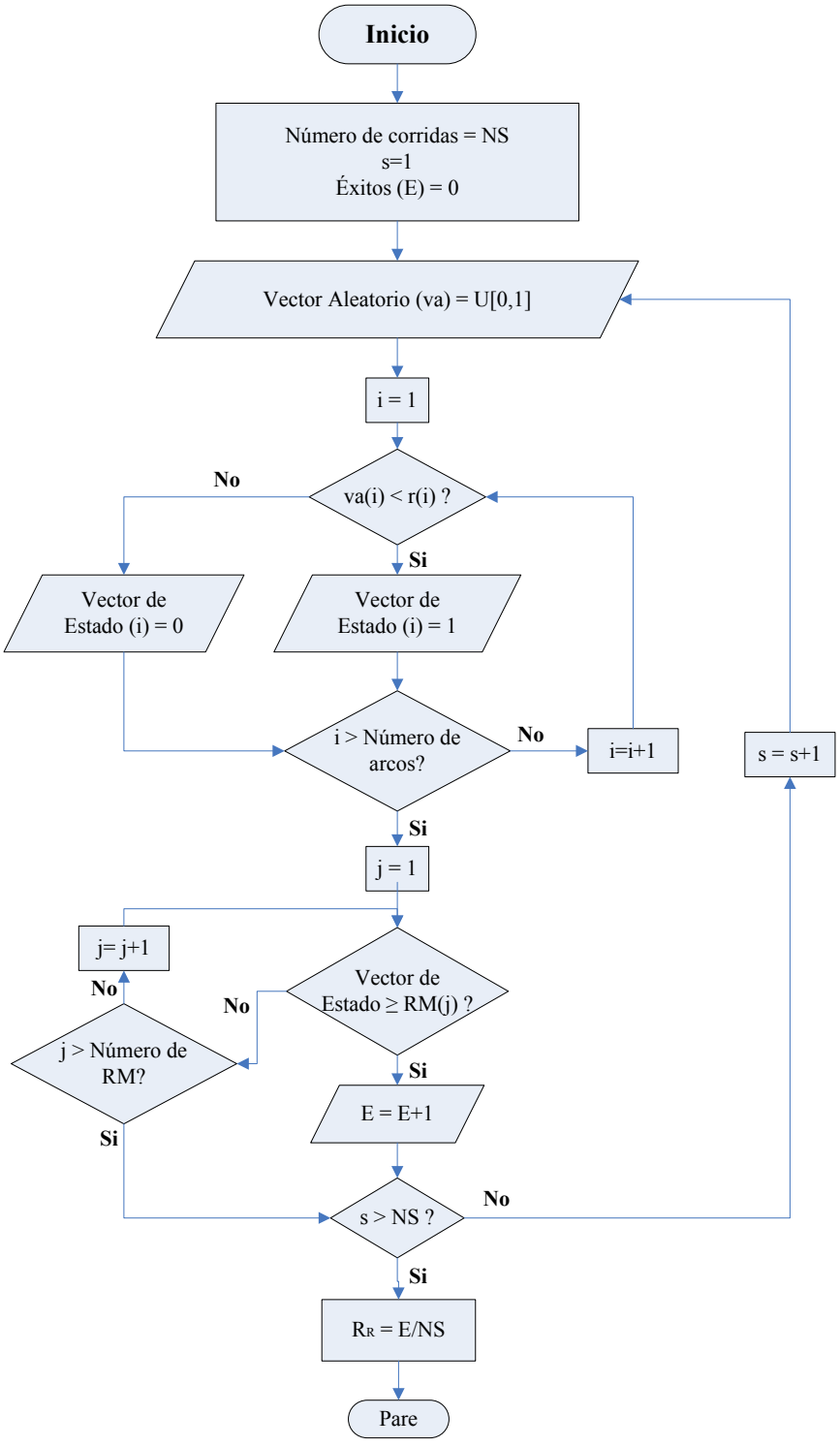


Figura 15. Diagrama de flujo para la Simulación de Monte Carlo

El método es aplicable a cualquier tipo de problema, ya sea estocástico o determinístico. A diferencia de los métodos numéricos que se basan en evaluaciones en N puntos en un espacio M-dimensional para producir una solución aproximada, el método de Monte Carlo tiene un error absoluto en la estimación que es inversamente proporcional a $\sqrt{NS}$, siendo *NS* el número de corridas de simulaciones ejecutadas. Por lo tanto, la teoría sugiere que el número de corridas de simulación, *NS*, requeridas en función de la precisión, o error, del estimado es inversamente proporcional al cuadrado del error, esto quiere decir que si se desea estimar una confiabilidad de la red con cuatro posiciones decimales correctas (0.0001) se requieren alrededor de 100 millones de simulaciones.

3.1.2.1.1 Experimentos y determinación de parámetros

La precisión del estimado de confiabilidad de la simulación se ve afectado por el tamaño de la red (n) y la confiabilidad promedio asignada a los arcos (r_p), por consiguiente, para establecer el número de corridas de simulación (*NS*) adecuado para aproximar la confiabilidad de las redes considerando su tamaño, la confiabilidad promedio asociada a sus arcos y el error deseado, se corrieron dos experimentos factoriales completos con cinco réplicas con las siguientes características:

Primer Experimento:

Tabla 19. Factores y niveles del primer experimento

Factores	**Niveles**	
Tamaño de la red (n)	5, 6, 7	
Confiabilidad de los arcos	**Promedio (r_p)**	**Rango**
	0.5	[0 , 1]
	0.6	[0.2 , 1]
	0.7	[0.4 , 1]
	0.8	[0.6 , 1]
	0.9	[0.8 , 1]
	0.95	[0.9 , 1]
	0.975	[0.95 , 1]
	0.995	[0.99 , 1]
Número de simulaciones (*NS*)	(10, 100, 500, 1000, 5000, 10000) x 10^3	

Variables de respuesta:

1. Confiabilidad simulada de la red
2. Tiempo de cómputo de la confiabilidad exacta de la red

3. Tiempo de cómputo de la confiabilidad simulada de la red
4. Tiempo de cómputo del límite superior de la confiabilidad de la red

Segundo Experimento:

Tabla 20. Factores y niveles del segundo experimento

Factores	**Niveles**	
Tamaño de la red (*n*)	4, 5, 6, 7	
Confiabilidad de los arcos	**Promedio (r_p)**	**Rango**
	0.05	[0 , 0.1]
	0.2	[0 , 0.4]
Número de simulaciones (*NS*)	(7.5, 750, 75000, 7500000) x 10^3	

Variables de respuesta:

1. Confiabilidad simulada de la red
2. Tiempo de cómputo de la confiabilidad exacta de la red
3. Tiempo de cómputo de la confiabilidad simulada de la red
4. Tiempo de cómputo del límite superior de la confiabilidad de la red

Se tomaron los resultados del error (confiabilidad simulada – confiabilidad exacta) para los 840 tratamientos efectuados en los dos experimentos y se obtuvo un modelo de regresión lineal para predecir el error absoluto, |error|, considerando solamente la diferencia absoluta entre la confiabilidad simulada y la confiabilidad exacta, en función de *n*, r_p y *NS*.

Se seleccionó como variable de respuesta el logaritmo natural del valor absoluto de los errores (y = ln|error|). La selección de variables se hizo con un nivel de significancia del 5%; la Tabla 21 lista los coeficientes, errores estándar y valores p asociados a cada variable regresora seleccionada. En la Ecuación (3.7) se presenta el modelo de regresión obtenido con un R^2 de 81.3% y un MSE de 1.50 (se eliminaron 16 datos espurios).

Tabla 21. Errores y valores p de las variables regresoras del ln|error|

Variable	**Coeficiente**	**Error Estándar**	**Valor p**
Constante	-5.7272	0.3908	0.0000
n	-0.5795	0.0522	0.0000
NS	-0.1547×10^{-4}	1.81×10^{-6}	0.0000
r_p	2.2131	0.3190	0.0000
$1 / NS^2 r_p$	35.4285	4.837	0.0000
$1/ r_p^2 NS$	-0.1872	0.3134	0.0000
$\ln(NS) r_p$	0.7419	0.7863	0.0000
$\ln(NS) r_p^2$	-1.7888	0.0848	0.0000

$$\ln|error| = -5.7272 - 0.5795n - 0.00001547NS + 2.2131r_p + 35.4285\frac{1}{NS^2 r_p} - 0.1872\frac{1}{r_p{}^2 NS} + 0.7419\ln(NS)r_p - 1.7888\ln(NS)r_p{}^2 \quad (3.7)$$

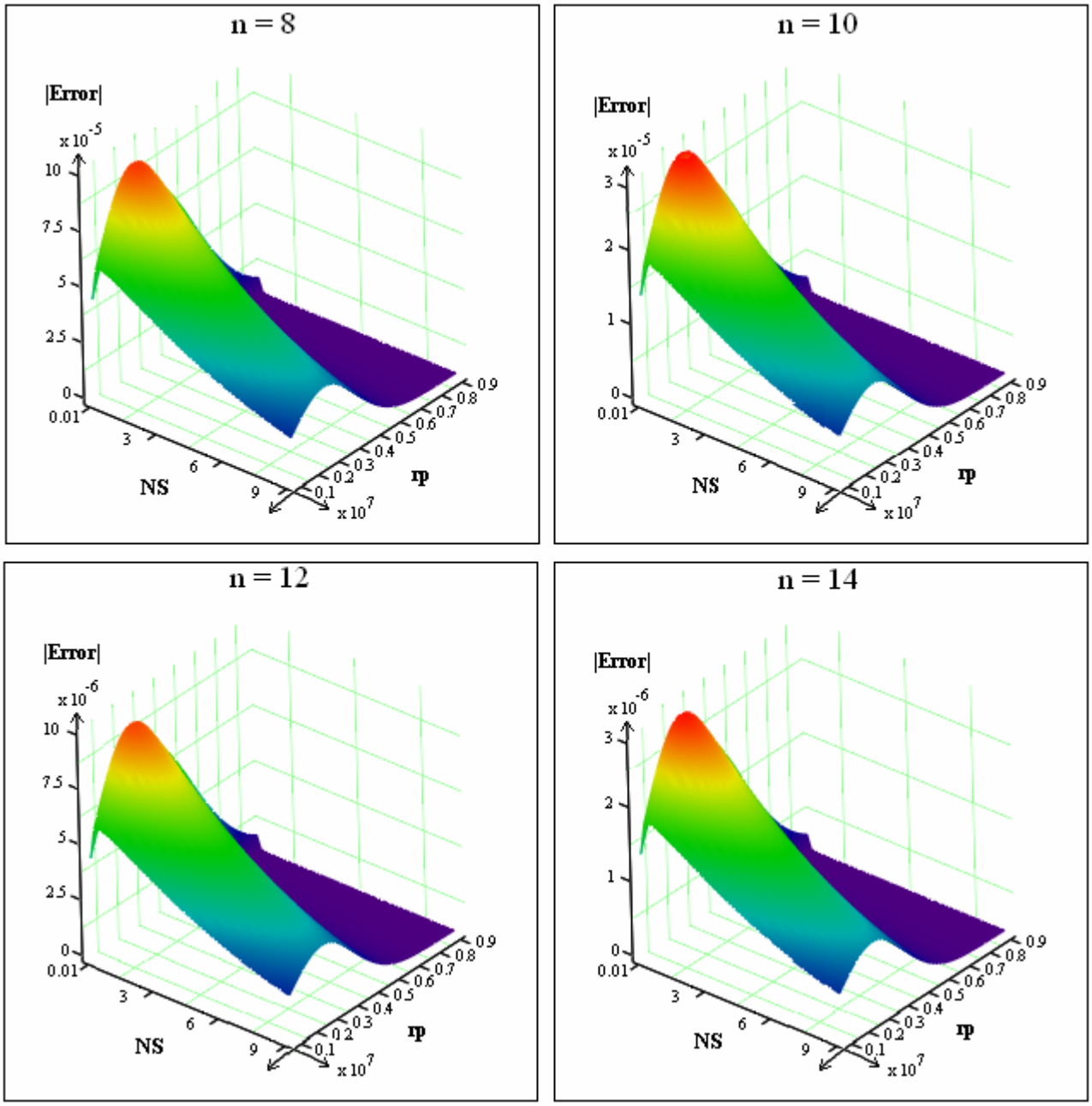


Figura 16. Superficie de respuesta del |error| para diferentes tamaños de red (n)

La Figura 16 presenta cuatro superficies de respuesta (correspondiente a cuatro tamaños de redes, n = 8, 10, 12 y 14) para el |error| predicho por el modelo de regresión en función de r_p y *NS*. De estas superficies puede concluirse que el |error| disminuye a medida que *NS* aumenta y el |error| es máximo para valores de r_p alrededor de 0.3, lo que se confirma observando las gráficas de la Figura 17 donde se ilustra el comportamiento del |error| predicho para las redes con 5, 6 y 7 nodos para confiabilidades promedio de arco (r_p) de 0.2, 0.3, 0.4 y 0.5.

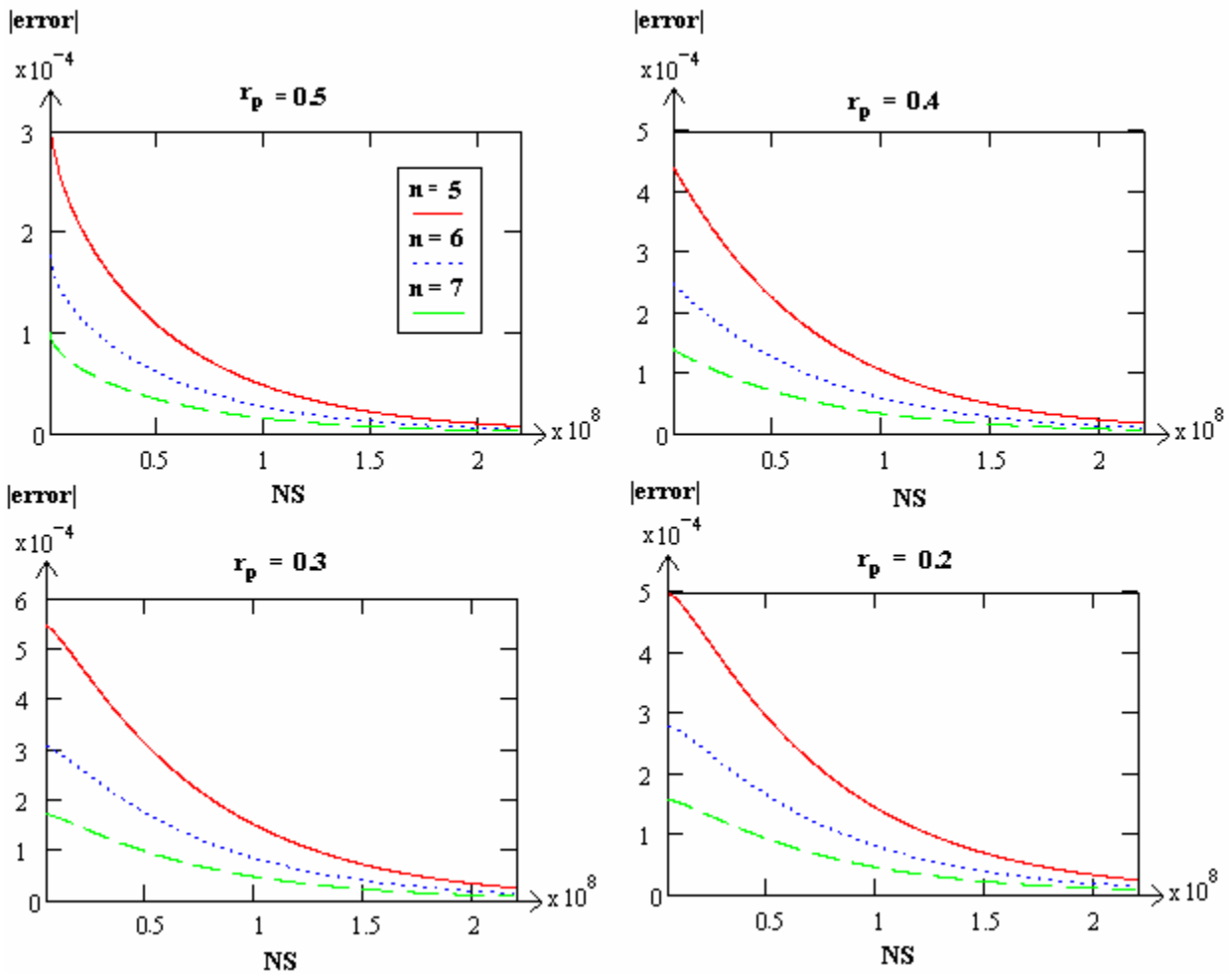


Figura 17. Gráficas de |error| predicho vs. *NS* para redes de 5, 6 y 7 nodos

De manera similar al |error|, se tomaron los 840 resultados obtenidos para el tiempo de simulación (t_s) y se obtuvo un modelo de regresión lineal para predecir el logaritmo natural de t_s (y = ln(t_s)) en función de n, r_p y *NS*. En la ecuación (3.8) se presenta el modelo de regresión obtenido con un R^2 del 99% y un MSE de 0.0034 (se eliminaron 12 datos espurios). La Tabla 22 lista los coeficientes, errores estándar y valores p asociados a cada variable regresora.

$$\ln(t_s) = -3.50863 + 0.0000008NS + 0.990648\ln(NS) + 9.0341\frac{1}{n^2} + 0.7312\frac{1}{r_p} - 0.0271\frac{1}{r_p^{\,2}} - 2.9123\frac{1}{r_p n} + 0.107871\frac{1}{r_p^{\,2} n} \quad (3.8)$$

Tabla 22. Errores y valores p de las variables regresoras del $\ln|t_s|$

Variable	Coeficiente	Error Estándar	Valor p
Constante	-3.5086	0.0175	0.0000
NS	8×10^{-7}	1×10^{-7}	0.0000
ln(NS)	0.9906	9.4×10^{-4}	0.0000
$1/n^2$	9.0341	0.5292	0.0000
$1/r_p$	0.7312	0.0192	0.0000
$1/r_p^{\,2}$	-0.0271	9.1×10^{-4}	0.0000
$1/r_p n$	-2.9123	0.1112	0.0000
$1/r_p^{\,2} n$	0.1079	0.0053	0.0000

La Figura 18 presenta la superficie de respuesta para el t_s predicho por el modelo de regresión para una red con 10 nodos en función de r_p y *NS*. De esta superficie puede concluirse que el t_s es creciente a medida que aumenta *NS*. La Figura 19 muestra la gráfica de NS vs. t_s predicho para redes con 5, 6 y 7 nodos donde se observa que a mayor *n* el t_s aumenta y esta diferencia es creciente cuando *NS* se hace grande.

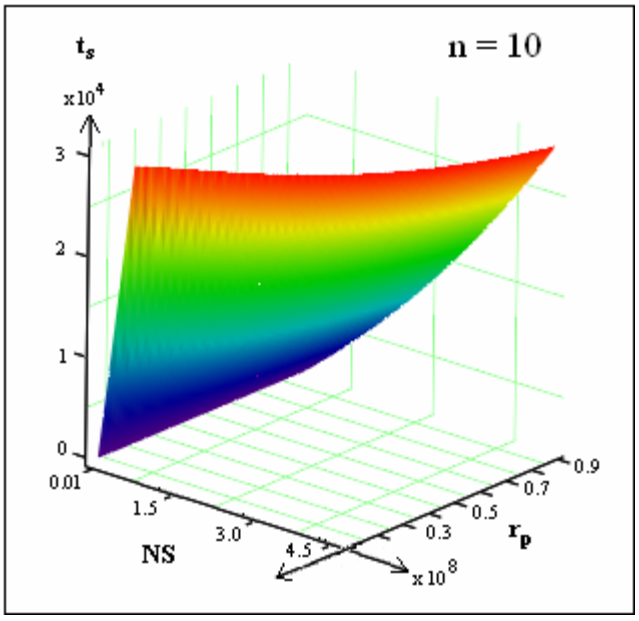


Figura 18. Superficie de respuesta del tiempo de simulación para 10 nodos

Como se enunció en la descripción del problema, se ha descartado trabajar con redes con arcos con confiabilidades mayores que 0.5 dado que, en estos casos, la confiabilidad de la red es cercana a 1 y la oportunidad de mejorarlas mediante adición de arcos, objetivo de esta tesis, es limitada. Considerando los resultados de predicción presentados de la regresión del |error| donde se concluyó que el |error| es máximo para valores de r_p alrededor de 0.3, se ha decidido que la confiabilidad promedio de los arcos de las redes a incluir en el estudio sea menor a 0.3 ($r_p < 0.3$).

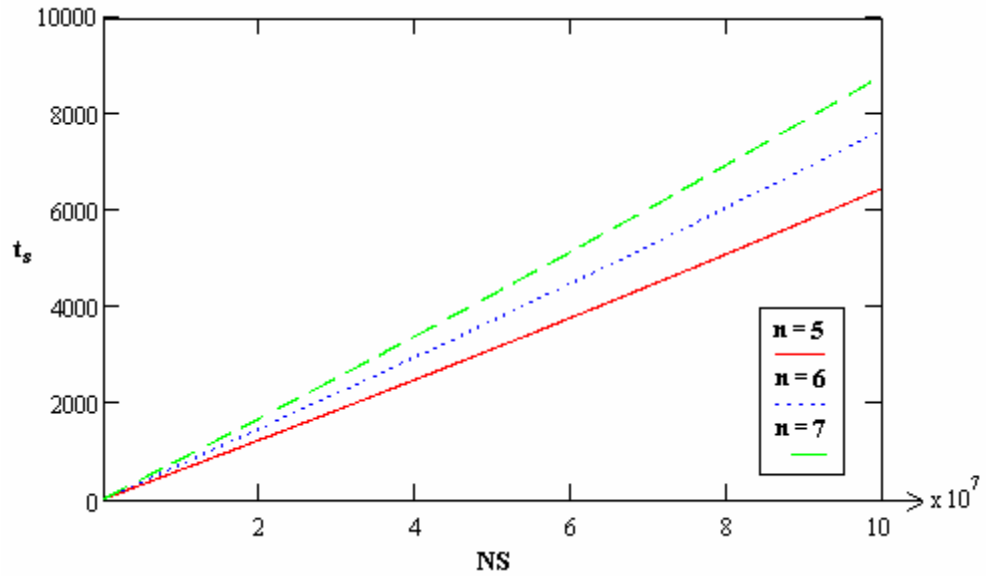


Figura 19. Gráfica de t_s predicho vs. *NS* para redes de 5, 6 y 7 nodos ($r_p = 0.3$)

La determinación del *NS* a emplear para cada red se encuentra sujeto al |error| de aproximación de confiabilidad esperado y al tiempo máximo establecido para cada simulación, t_s. En la Figura 20 se presenta la gráfica de |error| predicho vs. *NS* para redes de 8, 9 y 10 nodos con r_p de 0.3. En esta gráfica se observa que para obtener, por ejemplo, un |error| de aproximación por simulación de $3x10^{-5}$ se requiere efectuar 91.5, 52 y 6.5 millones de corridas de simulación para las redes de tamaño 8, 9 y 10 nodos, demandando al programa 148.6, 89.9 y 11.9 minutos, respectivamente. El t_s máximo que se estima aceptable para calcular la confiabilidad aproximada de una red dado el número de iteraciones que deben realizarse en la optimización es de 5 segundos y para ello tendría que establecerse un máximo de 51,000, 47,000 y 43,000 corridas de simulación (*NS*) para redes con 8, 9 y 10 nodos, respectivamente, con r_p de 0.3, disminuyendo el valor de *NS* a medida que r_p decrezca, aumentando el |error| a la cuarta y tercera posición decimal.

El modelo de regresión establecido para estimar el t_s se obtuvo a partir de los datos del experimento desarrollado con redes de 5, 6 y 7 nodos. Por lo tanto, con el fin de verificar la consistencia de predicción de t_s para redes con $r_p < 0.3$ y $10 < n \leq 20$ ($n = 20$ es el número máximo de tamaño de red para la que se ha obtenido el conjunto completo de rutas mínimas) se corrieron simulaciones con $NS = 100{,}000$ para cada tamaño de red entre 11 y 20 nodos. Los resultados obtenidos se presentan en la Tabla 23.

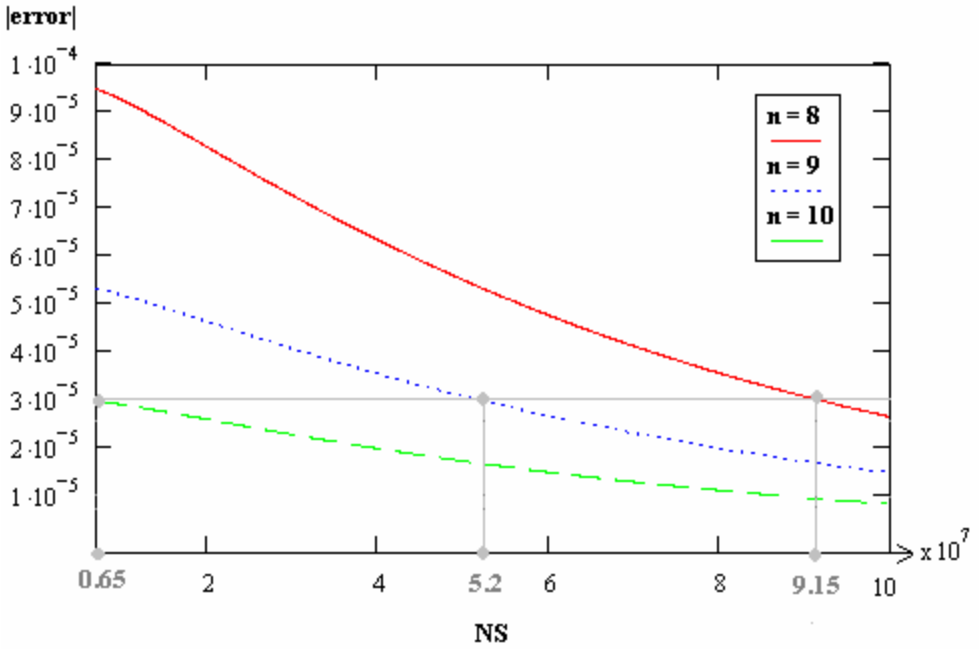


Figura 20. Gráfica de |error| predicho vs. *NS* para redes de 8, 9 y 10 nodos ($r_p = 0.3$)

Tabla 23. Resultados de t_s para simulaciones aplicando RM con 100,000 corridas ($10 < n \leq 20$)

n	$t_{s\,(min)}$	$t_{s\,(h)}$
11	7.8	0.13
12	15.6	0.26
13	33.9	0.57
14	54.7	0.92
15	101.6	1.69
16	221.4	3.69
17	408.85	6.81
18	1145.8	19.10
19	2187.5	36.46
20	3937.8	65.63

Con base en los resultados presentados en la Tabla 23, se establece que la SMC desarrollada a partir de la metodología de Ramírez y Coit (2005), combinada con el uso de rutas mínimas para determinar si la red funciona o no, es utilizable para redes con un máximo de 12 nodos por el tiempo requerido para estimar la confiabilidad de la red con 100,000 corridas de simulación.

3.1.2.2 Estimación de la confiabilidad empleando el Algoritmo de Dijkstra

Se propone una metodología de simulación de Monte Carlo alterna para poder calcular la confiabilidad aproximada de redes con *n* mayor de 12 en menor tiempo (Tabla 24), empleando el algoritmo de Dijkstra (Dijkstra, 1959), AD, para estimar los éxitos de cada corrida de simulación a partir de la determinación de la existencia de conexión entre los nodos origen y destino para una matriz de distancia generada aleatoriamente.

Tabla 24. Resultados de t_s para simulaciones aplicando AD con 100,000 corridas ($10 < n \leq 20$)

n	t_s (seg)	t_s (min)
11	5.11	0.0852
12	6.25	0.1042
13	7.42	0.1237
14	8.74	0.1456
15	9.67	0.1612
16	10.94	0.1823
17	11.13	0.1855
18	11.59	0.1932
19	12.15	0.2025
20	12.50	0.2083

A continuación se describe el procedimiento seguido para calcular la confiabilidad de una red compleja aplicando el AD en la SMC:

Paso 1: Se genera una matriz de tamaño *n* x *n* con valores aleatorios uniformes entre 0 y 1.

Paso 2: Se construye una matriz de distancias a partir de la comparación de cada posición de la matriz aleatoria, correspondiente a un posible arco de la red, con la confiabilidad asociada al arco, si el valor aleatorio es menor que la confiabilidad del arco se ubica un 1 en la posición del arco dentro de la matriz de distancia, de lo contrario se asigna ∞. El 1 en la matriz de distancias indica que existe un arco que mide una unidad conectando el nodo que

corresponde a la fila con el nodo que corresponde a la columna, y el ∞ significa que no hay conexión entre los nodos.

Paso 3: Se aplica el AD para determinar, a partir de la matriz de distancia, el tamaño (número de arcos) de la ruta más corta entre el nodo 1 (origen) y el nodo n (destino). Si la ruta más corta es menor o igual al número de nodos de la red (*n*) se establece que hay conexión entre los nodos origen y destino y por lo tanto se consideraría como un éxito.

Paso 4: Cada iteración de los pasos 1-3 es denominada una corrida de simulación. Se realizan *NS* corridas de simulación y la proporción de éxitos en las *NS* corridas es el valor estimado de la confiabilidad.

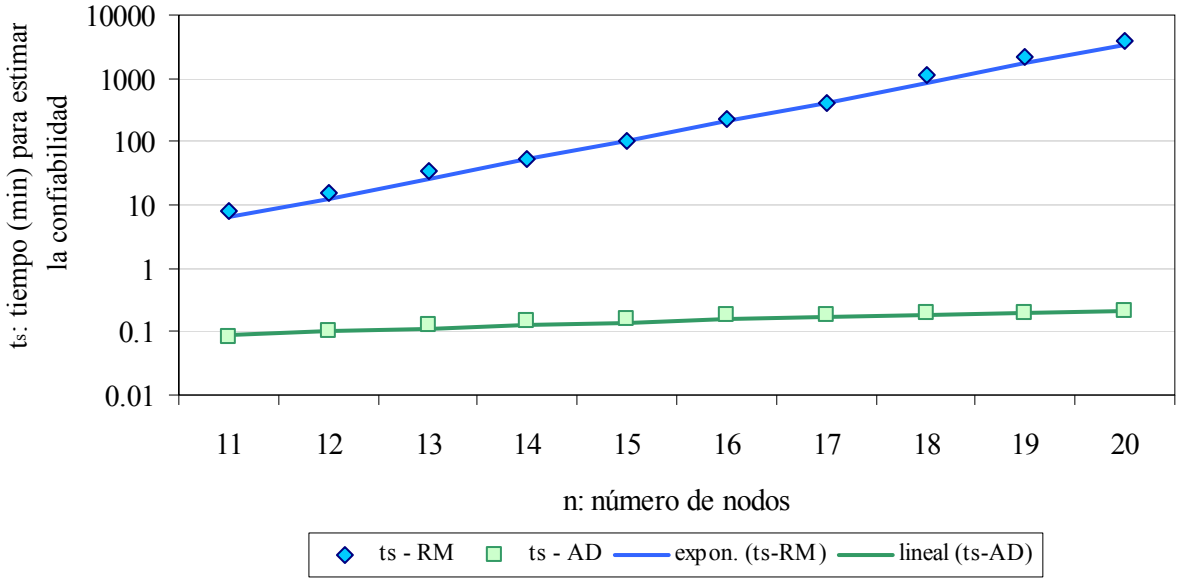


Figura 21. Tiempo para estimar la confiabilidad vs. tamaño de la red

En la Figura 21 se observa que el tiempo de simulación (t_s), en escala logarítmica, requerido para estimar la confiabilidad de las redes empleando la SMC con el método de Rutas Mínimas (SMCRM) crece exponencialmente a medida que *n* aumenta, a diferencia del crecimiento lineal en *n* observado para el tiempo de simulación (t_s) requerido empleando SMC con el Algoritmo de Dijkstra (SMCAD).

Al inicio de esta investigación se proponía utilizar en la optimización de las redes valores exactos de la confiabilidad, dado que se presentaron limitantes computacionales obteniendo

resultados exactos para redes con máximo 7 nodos (a = 21 arcos) se propuso la alternativa de estimar un valor aproximado de la confiabilidad de las redes empleando Simulación de Monte Carlo determinando el funcionamiento de la red (éxito) a partir de la comparación de un vector binario con las rutas mínimas de la red. Esta alternativa, denominada SMCRM, generó resultados satisfactorios para redes con máximo 12 nodos (a = 66 arcos). Con el fin de poder estimar la confiabilidad de redes de mayor tamaño, se propone utilizar la metodología de Simulación de Monte Carlo empleando el algoritmo del camino más corto conocido como Algoritmo de Dijkstra, llegando a obtener resultados en un tiempo de simulación razonable para redes con máximo 20 nodos (a = 190 arcos).

En conclusión, las redes que se van a trabajar tendrán entre 5 y 20 nodos, la confiabilidad de las redes con menos de 12 nodos serán aproximadas empleando la SMCRM, la confiabilidad de las redes con más de 12 nodos serán aproximadas con la metodología de SMCAD y la confiabilidad promedio de sus arcos será menor a 0.3 (r_p < 0.3). El número de corridas de simulación, *NS*, se encuentra sujeto al tiempo total de ejecución de los algoritmos, por lo tanto, su determinación estará basada en el número de iteraciones del algoritmo y el error máximo de aproximación por simulación establecido. En la siguiente sección se explica con detalle el procedimiento a seguir para el cálculo de *NS*.

3.2 Optimización

Dos algoritmos de optimización heurística son usados y comparados para determinar el número y la ubicación óptima de los arcos adicionales que incrementan la confiabilidad de la red: un algoritmo genético (AG) y un algoritmo basado en programación lineal entera secuencial (PLES) planteado por el doctor Noel Artiles y desarrollado y programado por la autora.

Los algoritmos son ilustrados empleando los diagramas de red (Figura 6 y Figura 7), matriz de conexión, datos de confiabilidad (Tabla 1 y Tabla 2), costos asociados a cada arco redundante (Tabla 3), y presupuesto de inversión (P = $99,000), utilizados en el ejemplo presentado en el Capítulo 2 de este trabajo.

3.2.1 Algoritmo Genético

Un algoritmo genético, AG, es un método general para resolver problemas de "búsqueda de soluciones" (Mitchell, 1998). El objetivo es encontrar eficientemente una solución para un problema en un espacio grande de soluciones candidatas.

El algoritmo inicia con un conjunto de soluciones llamado población y cada solución candidata es representada por un cromosoma. Un AG básico es descrito a continuación de manera general:

Paso 1. Inicio. Generación de una población aleatoria de *c* cromosomas.

Paso 2. Función de aptitud. Evaluación de la función de aptitud (conocida en la literatura en inglés como función "fitness") para cada cromosoma en la población.

Paso 3. Nueva población. Creación de una nueva población a partir de la repetición de los siguientes pasos hasta que la nueva población esté completa.

a. Seleccionar dos cromosomas padres de una población de acuerdo a su aptitud (mejor aptitud indica mayor opción de ser seleccionado).
b. Mezcla. Con una probabilidad de mezcla establecida los padres son cruzados para formar dos nuevos descendientes (hijos). Los hijos deben ser una combinación de sus padres.
c. Mutación. Con base en la probabilidad de mutación los hijos son modificados en su composición genética (organización o ubicación de los genes dentro del cromosoma).
d. Ubicación de los cromosomas generados en la nueva población.

Paso 4. Selección. Basado en su aptitud, seleccionar los *c* cromosomas para formar una nueva población.

Paso 5. Prueba. Si el criterio de terminación se cumple, parar, y se presenta la mejor solución en la población actual. De lo contrario, retornar al paso 2.

En el AG propuesto en esta tesis, la representación genética de un cromosoma $\mathbf{y_c}$ es

$\mathbf{y_c} =$

y_{c1}	y_{c2}	y_{c3}	y_{c4}	y_{c5}	…	y_{ca}

donde y_{ci}: número de arcos tipo i adicionales asignados a la red, i = 1, 2,…, *a*. Por lo tanto, cada cromosoma de la población va a representar un posible vector solución $\mathbf{y_c}$ conformado por *a*

genes que indican el número de arcos a adicionar a la red original. Los cromosomas van a estar codificados con números enteros que pueden variar en un rango:

rango = $[\mathbf{0}\ ;\ \mathbf{LS}]$

$LS_i = maxred_i$ - yo_i, i = 1, 2, ..., a

$maxred_i$: número máximo de arcos tipo i en la red

yo: vector de estado de la red original

Después de determinar la representación genética de las soluciones potenciales es necesario definir el procedimiento para crear la población inicial de soluciones.

3.2.1.1 Población Inicial

La población inicial se determina mediante una selección aleatoria de c cromosomas, donde c es el tamaño de la población. Previas experimentaciones en el campo (Coit y Smith, 1995) han determinado que un tamaño de población de 40 cromosomas converge rápidamente y genera buenas soluciones. No obstante, la teoría sugiere que el tamaño de la población, c, crezca con el tamaño del problema, por lo tanto, se ha determinado que c sea el producto entre a y el valor máximo de **maxred** ($c = a$(max(**maxred**))).

3.2.1.2 Función de Aptitud

La función de aptitud f(**yo**,**y**) para cada cromosoma de la población es definido como la función objetivo menos una función de penalidad por incumplimiento de la restricción de presupuesto.

Notación:

$R_{RT}(\mathbf{yo},\mathbf{y})$: confiabilidad total de la red con arcos adicionales

$Pen(\mathbf{y})$: función de penalidad

$C(\mathbf{y})$: costo total del vector solución **y**

a: número de posibles combinaciones de pares de nodos $\left(a = n(n\text{-}1)/2\right)$

C_i: costo de adicionar a la red un arco tipo i, i = 1, 2,..., a

P: presupuesto total disponible

yo: vector de estado de la red original

n: número de nodos de la red

Función de Aptitud: $f(\mathbf{yo},\mathbf{y}) = R_{RT}(\mathbf{yo},\mathbf{y}) - Pen(\mathbf{y})$

Función de Penalidad: $Pen(\mathbf{y}) = \max\left(0\ ,\ \frac{C(\mathbf{y}) - P}{P}\right)$ donde $C(\mathbf{y}) = \sum_{i=1}^{a} C_i y_i$

3.2.1.3 Selección

La selección en general es una consecuencia de la competencia entre individuos de la población. Ésta se refiere a la manera en que los individuos son seleccionados para formar la nueva población después de cada generación. El método de selección usado en el AG propuesto es la selección de torneo. Cuatro cromosomas son seleccionados aleatoriamente y el de mayor aptitud es seleccionado como un padre.

3.2.1.4 Operadores Genéticos

El AG propuesto involucra dos tipos de operadores genéticos: mutación y mezcla. Los mejores 5 cromosomas de cada generación pasan a la siguiente generación sin ser modificados, a esta evolución se le llama elitismo. No obstante, estos cromosomas participan en el proceso de selección para ser padres.

El primer operador genético es la mutación. La mutación usada es una mutación uniforme de dos pasos, primero el algoritmo selecciona una fracción del cromosoma para mutación con base en una tasa de mutación del 25% lo que indica que cada gen del cromosoma tiene una probabilidad del 25% de ser seleccionado para mutar; por ejemplo, para una red de 5 nodos (10 tipos de arcos) con restricción máxima de dos arcos por cada tipo de arco (**LS** = [2, 2, 1, 1, 2, 1, 2, 1, 1, 2]) se seleccionan para mutación los genes 2, 4, 6 y 7 del cromosoma $\mathbf{y_c}$

$\mathbf{y_c}$ =

1	1	2	1	0	1	0	2	1	1

luego, cada gen seleccionado es reemplazado por un valor aleatorio (redondeado para asegurar que el gen sea entero) seleccionado uniformemente dentro del rango de cada gen; para el ejemplo, los valores aleatorios seleccionados son 2, 0, 1 y 2, el nuevo cromosoma $\mathbf{y_c}$ mutado es

$\mathbf{y_c}$ =

1	2	2	0	0	1	2	2	1	1

El segundo operador genético es la mezcla. La mezcla se realiza entre dos padres que son seleccionados en función de su aptitud como se explicó en la sección anterior. El tipo de mezcla seleccionado es el de la mezcla dispersa, el algoritmo inicialmente genera un vector binario (**vb**), resultado de redondear un vector de números aleatorios uniformes entre 0 y 1 de tamaño *a*; continuando con el ejemplo de una red de 5 nodos ($a = 10$)

vb =	1	0	1	1	1	0	0	1	0	1

luego, se seleccionan del padre los genes de las posiciones correspondientes a 1's en el vector binario:

Padre =	1	2	0	1	2	1	2	0	0	1

se elige de la madre los genes de las posiciones correspondientes a 0's en el vector binario:

Madre =	2	1	1	1	1	1	1	1	1	2

y, finalmente, se combinan los genes seleccionados de los padres para formar el hijo:

$\mathbf{y_c}$ =	1	1	0	1	2	1	1	0	1	1

3.2.1.5 Criterios de Terminación

Se han establecido cuatro criterios de terminación del AG propuesto; cuando el algoritmo cumple con uno de estos criterios se presenta como respuesta la mejor solución de la población actual. Los criterios son:

a. Número total de generaciones (GenT). Se especifica que el número máximo de generaciones (iteraciones) que realiza el AG es 50.
b. Número de generaciones sin mejora de la función objetivo (GenFO). Se define que el algoritmo se detiene si no se presenta mejoría en la función objetivo por 10 generaciones consecutivas.

c. Tiempo (TT). El algoritmo se detiene si su función objetivo no mejora por 2,400 segundos (40 minutos) o si han transcurrido 86,400 segundos (24 horas) desde su inicialización.

d. Soluciones repetidas (SR). El AG en sus 10 primeras iteraciones estimará las confiabilidades de las redes ejecutando el número de corridas de simulación, *NS*, mínimas requeridas para que la SMC aproxime el valor con un error máximo de 0.1. Para las siguientes 10 iteraciones el *NS* aumentará al número necesario para que la aproximación sea con un error máximo de 0.01. A partir de la iteración 21 el error máximo esperado será de 0.001 y desde la iteración 31 el error máximo disminuirá a 0.0001 unidades. Se establece que el algoritmo se detendrá si se han ejecutado más de 20 iteraciones (error por aproximación de 0.001) y ha encontrado el mismo vector solución (igual asignación de cantidad y ubicación de los arcos en redundancia - SR) durante 5 iteraciones consecutivas.

   Dado que los resultados de las corridas de simulación dentro de la SMC siguen una distribución Bernoulli (1 si la red funciona, 0 de lo contrario) cuya suma dividida por *NS* es la confiabilidad aproximada de la red, se deduce que el *NS* para una iteración del AG en la que se desea un error máximo, *error*, para una confiabilidad R_{RT} (dada o estimada en la iteración anterior) y con una confianza del 95%, está dado por

$$NS = 4R_{RT}(1 - R_{RT})(error^{-2}) \tag{3.9}$$

El AG es programado en Matlab 7.0.1. empleando como base de programación las funciones de la herramienta de algoritmos genéticos (Genetic Algorithm Direct Search Toolbox 1.0.2), y calculando la confiabilidad de las redes por simulación de Monte Carlo, SMC. De la ejecución de la rutina en Matlab, ingresando la información del ejemplo, se extraen los resultados presentados en la Tabla 25 (confiabilidad calculada empleando SMCRM).

Para la red ejemplo, el algoritmo se detiene en la iteración número 22 por alcanzar el número de soluciones repetidas (SR) establecido, por 5 generaciones consecutivas obtuvo el mismo vector solución. El resultado de la optimización es el siguiente:

yr = **yo** + **y*** = [2, 2, 1, 1, 2, 2, 2, 2, 2, 2] , donde **y*** es la solución óptima al problema

R_{RT} = 0.8534

C = \$83,312

La confiabilidad de la red original era 10.29% y con la adición de 11 arcos redundantes (**y*** = [1, 1, 0, 0, 1, 1, 1, 2, 2, 2]) se aumenta en un 75.05% alcanzando el 85.34% de confiabilidad (|error| = 0.0003) y con una inversión de $83,312 que se encuentra dentro del presupuesto indicado. El resultado de optimización obtenido empleando AG con SMC es igual al alcanzado por el Solver de Excel utilizando el cálculo exacto para la confiabilidad, presentado en el Capítulo 2.

Tabla 25. Resultados por iteración del Algoritmo Genético

Generación	y	Yr	Mejor Aptitud	Aptitud Promedio	Costo
1	[0, 1, 0, 1, 1, 0, 1, 2, 2, 1]	[1, 2, 1, 2, 2, 1, 2, 2, 2, 1]	0.8020	0.6422	$ 80,107
2	[0, 0, 0, 1, 1, 1, 1, 2, 2, 2]	[1, 1, 1, 2, 2, 2, 2, 2, 2, 2]	0.8410	0.7073	$ 94,607
3	[0, 0, 0, 1, 1, 1, 1, 2, 2, 2]	[1, 1, 1, 2, 2, 2, 2, 2, 2, 2]	0.8410	0.7517	$ 94,607
4	[0, 0, 0, 1, 1, 1, 1, 2, 2, 2]	[1, 1, 1, 2, 2, 2, 2, 2, 2, 2]	0.8410	0.7415	$ 94,607
5	[1, 0, 0, 0, 1, 1, 1, 2, 2, 2]	[2, 1, 1, 1, 2, 2, 2, 2, 2, 2]	0.8510	0.7610	$ 82,741
6	[1, 0, 0, 0, 1, 1, 1, 2, 2, 2]	[2, 1, 1, 1, 2, 2, 2, 2, 2, 2]	0.8510	0.7616	$ 82,741
7	[1, 0, 0, 0, 1, 1, 1, 2, 2, 2]	[2, 1, 1, 1, 2, 2, 2, 2, 2, 2]	0.8510	0.7342	$ 82,741
8	[1, 0, 0, 0, 1, 1, 1, 2, 2, 2]	[2, 1, 1, 1, 2, 2, 2, 2, 2, 2]	0.8510	0.7527	$ 82,741
9	[1, 0, 0, 0, 1, 1, 1, 2, 2, 2]	[2, 1, 1, 1, 2, 2, 2, 2, 2, 2]	0.8510	0.7896	$ 82,741
10	[1, 1, 0, 0, 1, 1, 1, 2, 2, 2]	[2, 2, 1, 1, 2, 2, 2, 2, 2, 2]	0.8580	0.7120	$ 83,312
11	[1, 0, 0, 0, 1, 1, 1, 2, 2, 2]	[2, 1, 1, 1, 2, 2, 2, 2, 2, 2]	0.8549	0.8148	$ 82,741
12	[1, 1, 0, 0, 1, 1, 1, 2, 2, 2]	[2, 2, 1, 1, 2, 2, 2, 2, 2, 2]	0.8553	0.7215	$ 83,312
13	[1, 1, 0, 0, 1, 1, 1, 2, 2, 2]	[2, 2, 1, 1, 2, 2, 2, 2, 2, 2]	0.8553	0.7996	$ 83,312
14	[1, 1, 0, 0, 1, 1, 1, 2, 2, 2]	[2, 2, 1, 1, 2, 2, 2, 2, 2, 2]	0.8553	0.7995	$ 83,312
15	[1, 1, 0, 0, 1, 1, 1, 2, 2, 2]	[2, 2, 1, 1, 2, 2, 2, 2, 2, 2]	0.8553	0.8289	$ 83,312
16	[1, 1, 0, 0, 1, 1, 1, 2, 2, 2]	[2, 2, 1, 1, 2, 2, 2, 2, 2, 2]	0.8553	0.8061	$ 83,312
17	[1, 1, 0, 0, 1, 1, 1, 2, 2, 2]	[2, 2, 1, 1, 2, 2, 2, 2, 2, 2]	0.8553	0.7438	$ 83,312
18	[1, 1, 0, 0, 1, 1, 1, 2, 2, 2]	[2, 2, 1, 1, 2, 2, 2, 2, 2, 2]	0.8553	0.7936	$ 83,312
19	[1, 1, 0, 0, 1, 1, 1, 2, 2, 2]	[2, 2, 1, 1, 2, 2, 2, 2, 2, 2]	0.8553	0.7758	$ 83,312
20	[1, 1, 0, 0, 1, 1, 1, 2, 2, 2]	[2, 2, 1, 1, 2, 2, 2, 2, 2, 2]	0.8553	0.8060	$ 83,312
21	[1, 1, 0, 0, 1, 1, 1, 2, 2, 2]	[2, 2, 1, 1, 2, 2, 2, 2, 2, 2]	0.8534	0.7805	$ 83,312
22	[1, 1, 0, 0, 1, 1, 1, 2, 2, 2]	[2, 2, 1, 1, 2, 2, 2, 2, 2, 2]	0.8534	0.8265	$ 83,312

La Figura 22 presenta el comportamiento de la función de aptitud a través de las generaciones. El valor de la mejor aptitud obtenida para la iteración 10 es calculada nuevamente para un mayor número de *NS* con el fin de reducir el error a 0.01 y éste valor es el que se

compara con la mejor aptitud obtenida en la iteración 11 para determinar cuál es la solución que continua como ganadora. Por este motivo se observa en la gráfica una disminución del valor de la función en la iteración 11 consecuencia de que el algoritmo encontró un mejor vector solución cuya función de aptitud fue estimada con menor error.

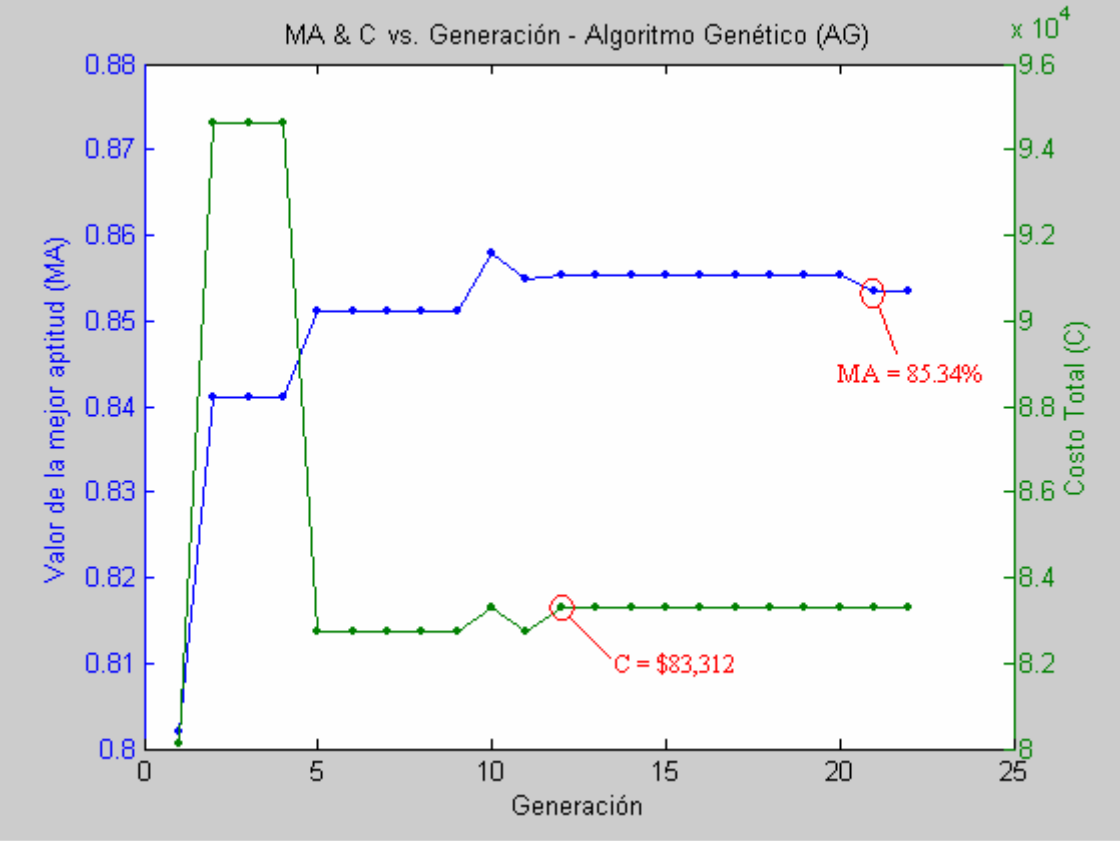


Figura 22. Confiabilidad de la red (aptitud) y costo asociado calculados en cada iteración

3.2.2 Algoritmo de Programación Lineal

El problema de optimización de una red compleja mediante adición de arcos redundantes es un problema de programación no lineal entero y por ello los métodos exactos para obtener la solución óptima al problema exigen mucho tiempo computacional y usualmente requieren gran cantidad de memoria de computadora (Kuo y Prassad, 2000). Por estas razones, los investigadores en optimización de confiabilidad se han enfocado en el desarrollo de aproximaciones heurísticas.

En búsqueda de una alternativa de optimización se desarrolla y propone un algoritmo basado en programación lineal entera secuencial, PLES. Las confiabilidades de las redes serán estimadas empleando la metodología de Simulación de Monte Carlo, SMC.

Paso 1: Se construye una matriz $\mathbf{Y_R}$ con a+2 filas por a columnas. La primera fila corresponde al vector de estado de la red original **yo**, la segunda fila corresponde a **yo** más un arco adicional tipo 1, la tercera fila corresponde a **yo** más un arco adicional tipo 2, y así sucesivamente hasta que la fila a+1 corresponde a **yo** más un arco adicional tipo a. La última fila (a+2) corresponde al vector de máxima redundancia, **maxred**.

Paso 2: Se construye un vector columna **R** de tamaño a+2 resultado del cálculo (por simulación) de las confiabilidades asociadas a cada fila de la matriz $\mathbf{Y_R}$.

Paso 3: Se determina un vector de β's resultado del ajuste de una ecuación de regresión para $\mathbf{R} = \mathbf{Y_R}\boldsymbol{\beta} + \boldsymbol{\varepsilon}$.

Paso 4: Se resuelve el problema de programación lineal entera, PLE:

$$\text{Maximizar } R_{\text{temp}}(\mathbf{yr}) = \sum_{i=1}^{a} \beta_i yr_i + \beta_0$$

Sujeto a:

$$yo_i \le yr_i \le \min(yo_i + 1, maxred_i)$$

$$\sum_{i=1}^{a} C_i yr_i \le P + \sum_{i=1}^{a} C_i yo_i \ ^*$$

donde:

yr : vector para la nueva red con arcos adicionales (red original más arcos redundantes)

$R_{\text{temp}}(\mathbf{yr})$: estimado de la confiabilidad para la red con arcos redundantes

* En la restricción de presupuesto se adiciona el costo de la red original al presupuesto dado ya que la variable de decisión del PLE es **yr** y ésta considera los arcos de la red original (**yo**) más los arcos redundantes (**y**).

Paso 5: Se calcula la confiabilidad R_{RTn} para la nueva red con vector de arcos **yr***, donde **yr*** es la solución óptima al problema descrito en el paso 4.

Paso 6: Se construye una matriz $\mathbf{YN_R}$ de tamaño 2a+2. $\mathbf{YN_R}$ es equivalente a la matriz $\mathbf{Y_R}$ reemplazando en su construcción a **yo** por **yr*** y adicionando a filas que corresponden a **yr*** menos un arco tipo 1 en la fila a+2, **yr*** menos un arco tipo 2 en la fila a+3, continuando la secuencia hasta **yr*** menos un arco tipo a en la fila 2a+1. La fila 2a+2 corresponde al vector

de máxima redundancia, **maxred**. Se eliminan las filas de $\mathbf{YN_R}$ que contengan vectores que incumplen la restricción de redundancia máxima por arco establecida, presentan números negativos o eliminan arcos existentes en la red original.

Paso 7: Se construye un vector columna **RN** resultado del cálculo de las confiabilidades asociadas a cada fila de la matriz **RN**.

Paso 8: : Se determina un vector de β's resultado del ajuste de una ecuación de regresión para $\mathbf{RN} = \mathbf{YN_R}\boldsymbol{\beta} + \boldsymbol{\varepsilon}$.

Paso 9: Se resuelve el problema de programación lineal entera, PLE:

$$\text{Maximizar } R_{\text{temp}}(\mathbf{ynr}) = \sum_{i=1}^{a} \beta_i ynr_i + \beta_0$$

Sujeto a:

$$\max(yo_i, yr_i - 1) \le yr_i \le \min(yr_i + 1, maxred_i)$$

$$\sum_{i=1}^{a} C_i ynr_i \le P + \sum_{i=1}^{a} C_i yo_i$$

donde:

$\mathbf{ynr}$: vector para la nueva red con arcos adicionales (red original más arcos redundantes)

$R_{\text{temp}}(\mathbf{ynr})$: estimado de la confiabilidad para la red con arcos redundantes

Paso 10: Se calcula la confiabilidad R_{RTn} para la nueva red con arcos adicionales **ynr***, donde **ynr*** es la solución óptima al problema de PLE del paso 9.

Paso 11: Se evalúan los criterios de terminación, si alguno se cumple, el algoritmo termina y se presenta la mejor solución. De lo contrario, se retorna al paso 6.

Criterios de terminación: se han establecido tres criterios de terminación para el algoritmo PLES; cuando el algoritmo cumple con uno de estos criterios se presenta como respuesta la mejor solución al problema actual. Los criterios son:

a. Número máximo de iteraciones (maxit). Se especifica que el número máximo de iteraciones totales que realizará el algoritmo es 50.
b. Número de iteraciones sin mejora de la función objetivo (maxitmejor). Se define que el número máximo de iteraciones que efectuará el algoritmo sin encontrar una solución que supere en confiabilidad a la mejor alternativa encontrada es 10.
c. Soluciones repetidas (maxigual). De manera consistente a la metodología empleada en el Algoritmo Genético propuesto, el algoritmo PLES en sus 10 primeras iteraciones ejecutará

NS corridas para aproximar la confiabilidad con un error máximo de 0.1. Para las siguientes 10 iteraciones el *NS* aumentará para que el error de aproximación sea máximo de 0.01. A partir de la iteración 21 el error máximo esperado será de 0.001 y desde la iteración 31 el error máximo disminuirá a 0.0001 unidades. Se establece que el algoritmo para si se han ejecutado más de 20 iteraciones y ha encontrado la misma solución durante 5 iteraciones.

Este algoritmo es programado en Matlab 7.0.1., calculando la confiabilidad de las redes con SMC. De la ejecución de la rutina en Matlab, ingresando la información del ejemplo, se extraen los resultados presentados a continuación (confiabilidad calculada empleando SMCRM).

1. Se establece que el número máximo de arcos en redundancia para esta red es 2 (**maxred** = [2, 2, 2, 2, 2, 2, 2, 2, 2, 2]).
2. Construcción del vector **yo** para la red original: **yo** = [1, 1, 1, 1, 1, 1, 1, 0, 0, 0]
3. Cálculo de R_{RO} (confiabilidad de la red original) mediante la fórmula del límite superior de la confiabilidad, LSC. El LSC de una red es el valor máximo que puede alcanzar la confiabilidad de la red. Se calcula asumiendo independencia de las rutas mínimas de la red y reemplazando directamente en la función de estructura de la red original ($\varphi(\mathbf{yo})$) las variables de estado asociadas a cada arco por su confiabilidad. Para el ejemplo, el LSC estaría definido por: $LSC = R_{RO} = 1 - [(1 - r_1 r_4)(1 - r_1 r_3 r_7)(1 - r_5 r_6 r_7)(1 - r_1 r_2 r_6 r_7)] = 0.1120$
4. Cálculo de *NS* para las primeras 10 iteraciones: $NS = 16 R_{RO}(1 - R_{RO})(error^{-2}) = 159.1 \approx 159$
5. Matriz $\mathbf{Y_R}$:

$$\mathbf{Y_R} = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 \\ 2 & 1 & 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 \\ 1 & 2 & 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 \\ 1 & 1 & 2 & 1 & 1 & 1 & 1 & 0 & 0 & 0 \\ 1 & 1 & 1 & 2 & 1 & 1 & 1 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 & 2 & 1 & 1 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 & 1 & 2 & 1 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 & 1 & 1 & 2 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 0 & 0 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 0 & 1 & 0 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 0 & 0 & 1 \\ 2 & 2 & 2 & 2 & 2 & 2 & 2 & 2 & 2 & 2 \end{bmatrix}$$

6. Vector de confiabilidades **R**:

R'= [0.1120,0.1975,0.1002,0.1152,0.1590,0.1335,0.1198,0.1287,0.2019,0.3728,0.1941,0.8769]

7. Determinación del vector de β's ($\mathbf{Y_R} = \mathbf{R\beta} + \boldsymbol{\varepsilon}$):

β'= [0.1120, 0.0641,-0.0332,-0.0182,0.0256,0.0001,-0.0136,-0.0047,0.0668,0.2377,0.0590]

8. Solución del problema de programación lineal entera, PLE:

$$\text{Maximizar } R_{\text{temp}}(\mathbf{yr}) = \sum_{i=1}^{a} \beta_i yr_i + \beta_0$$

$$\text{Sujeto a: } \sum_{i=1}^{a} C_i yr_i \le P + \sum_{i=1}^{a} C_i yo_i$$

yr*'=[2, 1, 1, 1, 2, 1, 1, 1, 1, 1]; $R_{temp}(\mathbf{yr^*}) = 0.5474$; $\sum_{i=1}^{10} C_i yr_i^* = \$\ 207{,}153 < \$\ 239{,}787$

9. Cálculo de R_{RTn} (confiabilidad de la nueva red): $R_{RTn} = \text{SMC}(NS, r, r_R) = 0.6261$

10. Matriz $\mathbf{YN_R}$:

$$\mathbf{YN_R} = \begin{bmatrix} 2&1&1&1&2&1&1&1&1&1 \\ 3&1&1&1&2&1&1&1&1&1 \\ 2&2&1&1&2&1&1&1&1&1 \\ 2&1&2&1&2&1&1&1&1&1 \\ 2&1&1&2&2&1&1&1&1&1 \\ 2&1&1&1&3&1&1&1&1&1 \\ 2&1&1&1&2&2&1&1&1&1 \\ 2&1&1&1&2&1&2&1&1&1 \\ 2&1&1&1&2&1&1&2&1&1 \\ 2&1&1&1&2&1&1&1&2&1 \\ 2&1&1&1&2&1&1&1&1&2 \\ 1&1&1&1&2&1&1&1&1&1 \\ 2&0&1&1&2&1&1&1&1&1 \\ 2&1&0&1&2&1&1&1&1&1 \\ 2&1&1&0&2&1&1&1&1&1 \\ 2&1&1&1&1&1&1&1&1&1 \\ 2&1&1&1&2&0&1&1&1&1 \\ 2&1&1&1&2&1&0&1&1&1 \\ 2&1&1&1&2&1&1&0&1&1 \\ 2&1&1&1&2&1&1&1&0&1 \\ 2&1&1&1&2&1&1&1&1&0 \\ 2&2&2&2&2&2&2&2&2&2 \end{bmatrix} \Rightarrow \text{corrección} \Rightarrow \mathbf{YN_R} = \begin{bmatrix} 2&1&1&1&2&1&1&1&1&1 \\ 2&2&1&1&2&1&1&1&1&1 \\ 2&1&2&1&2&1&1&1&1&1 \\ 2&1&1&2&2&1&1&1&1&1 \\ 2&1&1&1&2&2&1&1&1&1 \\ 2&1&1&1&2&1&2&1&1&1 \\ 2&1&1&1&2&1&1&2&1&1 \\ 2&1&1&1&2&1&1&1&2&1 \\ 2&1&1&1&2&1&1&1&1&2 \\ 1&1&1&1&2&1&1&1&1&1 \\ 2&1&1&1&1&1&1&1&1&1 \\ 2&1&1&1&2&1&1&0&1&1 \\ 2&1&1&1&2&1&1&1&0&1 \\ 2&1&1&1&2&1&1&1&1&0 \\ 2&2&2&2&2&2&2&2&2&2 \end{bmatrix}$$

11. Vector de confiabilidades **RN**:

RN'= [0.6261, 0.6325, 0.6363, 0.6830, 0.6348, 0.6726, 0.6493, 0.7375, 0.6946, 0.5839, 0.5712, 0.5862, 0.4578, 0.5236, 0.8773]

12. Determinación del vector de β's (**RN** = $\mathbf{YN_R}\,\boldsymbol{\beta} + \boldsymbol{\varepsilon}$):

β'= [0.1701, 0.0372, -0.0092, -0.0054, 0.0413, 0.0499, -0.0069, 0.0309, 0.0213, 0.1296, 0.0752]

13. Solución del problema de programación lineal entera, PLE:

$$\text{Maximizar } R_{\text{temp}}(\mathbf{ynr}) = \sum_{i=1}^{a} \beta_i ynr_i + \beta_0$$

$$\text{Sujeto a: } \sum_{i=1}^{a} C_i ynr_i \le P + \sum_{i=1}^{a} C_i yo_i$$

ynr*' = [2, 1, 1, 1, 2, 1, 2, 2, 2, 2]; $R_{\text{temp}}(\mathbf{ynr^*}) = 0.7982$; $\sum_{i=1}^{10} C_i ynr_i^* = \$215{,}361 < \$239{,}787$

y*' (**ynr*'** - **yo**) = [1, 0, 0, 0, 1, 0, 1, 2, 2, 2] ; $\sum_{i=1}^{10} C_i y_i^* = 74{,}574 < \text{P } (\$99{,}000)$

14. Inicialización del contador de iteraciones totales (debe ser menor que 50, número máximo de iteraciones establecido).
15. Cálculo de R_{RTn} (confiabilidad de la nueva red): R_{RTn} = SMC(NS, r, r_R) = 0.8359
16. Evaluación de criterios de parada:
    a. ¿La confiabilidad de la nueva red es mayor que la confiabilidad de la red en la anterior interacción?; respuesta: sí (0.8399 > 0.6261); por lo tanto, la confiabilidad de la nueva red es ahora la mejor confiabilidad. Se inicia el contador de mejor solución (debe ser menor de 10, número máximo de iteraciones sin mejorar la solución)
    b. ¿El número de iteraciones es mayor o igual a 50?; respuesta: no.
    c. ¿El número de iteraciones sin mejorar la respuesta es mayor o igual a 10?; respuesta: no.
17. Se retorna al paso 10 y continúa iterando hasta que se cumpla uno de los criterios de parada.

Para la red ejemplo, el algoritmo se detiene en la iteración número 22 por alcanzar el número máximo de iteraciones (5) con igual vector solución **yr**. La Tabla 26 presenta los resultados por iteración del algoritmo PLES.

El resultado de la optimización es el siguiente:

yr* = [2, 2, 1, 1, 2, 2, 2, 2, 2, 2]

R_{RT} = 0.8539

C = \$83,312

La confiabilidad de la red original era 10.29% y con la adición de 11 arcos redundantes (**y*** = [1, 1, 0, 0, 1, 1, 1, 2, 2, 2]) se aumenta en un 75.1% alcanzando el 85.39% de confiabilidad (|error| = 0.0002) y con una inversión de \$83,312 que se encuentra dentro del presupuesto indicado.

Tabla 26. Resultados por iteración del Algoritmo de Programación Lineal Entera Secuencial

Generación	y	yr	R_{RT}	Costo
1	[1, 0, 0, 0, 1, 0, 0, 1, 1, 1]	[2, 1, 1, 1, 2, 1, 1, 1, 1, 1]	0.6261	\$66,366
2	[1, 0, 0, 0, 1, 0, 1, 2, 2, 2]	[2, 1, 1, 1, 2, 1, 2, 2, 2, 2]	0.8359	\$74,574
3	[1, 1, 0, 0, 1, 0, 1, 2, 2, 2]	[2, 2, 1, 1, 2, 1, 2, 2, 2, 2]	0.8365	\$75,145
4	[1, 1, 0, 0, 1, 0, 1, 2, 2, 2]	[2, 2, 1, 1, 2, 1, 2, 2, 2, 2]	0.8365	\$75,145
5	[1, 1, 0, 0, 1, 1, 1, 2, 2, 2]	[2, 2, 1, 1, 2, 2, 2, 2, 2, 2]	0.8489	\$83,312
6	[1, 1, 0, 0, 1, 1, 1, 2, 2, 2]	[2, 2, 1, 1, 2, 2, 2, 2, 2, 2]	0.8489	\$83,312
7	[1, 1, 1, 0, 1, 0, 1, 2, 2, 2]	[2, 2, 2, 1, 2, 1, 2, 2, 2, 2]	0.8531	\$93,681
8	[1, 1, 1, 0, 1, 0, 1, 2, 2, 2]	[2, 2, 2, 1, 2, 1, 2, 2, 2, 2]	0.8531	\$93,681
9	[1, 1, 1, 0, 1, 0, 1, 2, 2, 2]	[2, 2, 2, 1, 2, 1, 2, 2, 2, 2]	0.8531	\$93,681
10	[1, 1, 1, 0, 1, 0, 1, 2, 2, 2]	[2, 2, 2, 1, 2, 1, 2, 2, 2, 2]	0.8531	\$93,681
11	[1, 1, 0, 0, 1, 1, 1, 2, 2, 2]	[2, 2, 1, 1, 2, 2, 2, 2, 2, 2]	0.8535	\$83,312
12	[1, 1, 0, 0, 1, 1, 1, 2, 2, 2]	[2, 2, 1, 1, 2, 2, 2, 2, 2, 2]	0.8535	\$83,312
13	[1, 1, 0, 0, 1, 1, 1, 2, 2, 2]	[2, 2, 1, 1, 2, 2, 2, 2, 2, 2]	0.8535	\$83,312
14	[1, 1, 0, 0, 1, 1, 1, 2, 2, 2]	[2, 2, 1, 1, 2, 2, 2, 2, 2, 2]	0.8535	\$83,312
15	[1, 1, 0, 0, 1, 1, 1, 2, 2, 2]	[2, 2, 1, 1, 2, 2, 2, 2, 2, 2]	0.8535	\$83,312
16	[1, 1, 0, 0, 1, 1, 1, 2, 2, 2]	[2, 2, 1, 1, 2, 2, 2, 2, 2, 2]	0.8535	\$83,312
17	[1, 1, 1, 0, 1, 0, 1, 2, 2, 2]	[2, 2, 2, 1, 2, 1, 2, 2, 2, 2]	0.8536	\$93,681
18	[1, 1, 0, 0, 1, 1, 1, 2, 2, 2]	[2, 2, 1, 1, 2, 2, 2, 2, 2, 2]	0.8537	\$83,312
19	[1, 1, 0, 0, 1, 1, 1, 2, 2, 2]	[2, 2, 1, 1, 2, 2, 2, 2, 2, 2]	0.8537	\$83,312
20	[1, 1, 0, 0, 1, 1, 1, 2, 2, 2]	[2, 2, 1, 1, 2, 2, 2, 2, 2, 2]	0.8537	\$83,312
21	[1, 1, 0, 0, 1, 1, 1, 2, 2, 2]	[2, 2, 1, 1, 2, 2, 2, 2, 2, 2]	0.8539	\$83,312
22	[1, 1, 0, 0, 1, 1, 1, 2, 2, 2]	[2, 2, 1, 1, 2, 2, 2, 2, 2, 2]	0.8539	\$83,312

La Figura 23 presenta el comportamiento de la función objetivo y el costo a través de las iteraciones, se observa una disminución en la confiabilidad en la iteración 21 consecuencia de la reevaluación de la confiabilidad de la red al aumentar *NS* para disminuir el error de aproximación por la simulación.

El resultado de optimización obtenido, número y ubicación de arcos redundantes, empleando el algoritmo propuesto PLES es igual al alcanzado por el Solver de Excel y el AG.

La diferencia entre los algoritmos de optimización en combinación con las metodologías de SMC radica en el tiempo de solución, para este ejemplo el AG con SMCRM demandó 1.15 minutos, el AG con SMCAD consumió 2.33 minutos, el algoritmo PLES con SMCRM tomó 0.97 minutos y el algoritmo PLES con SMCAD demandó 2.10 minutos en encontrar la respuesta óptima.

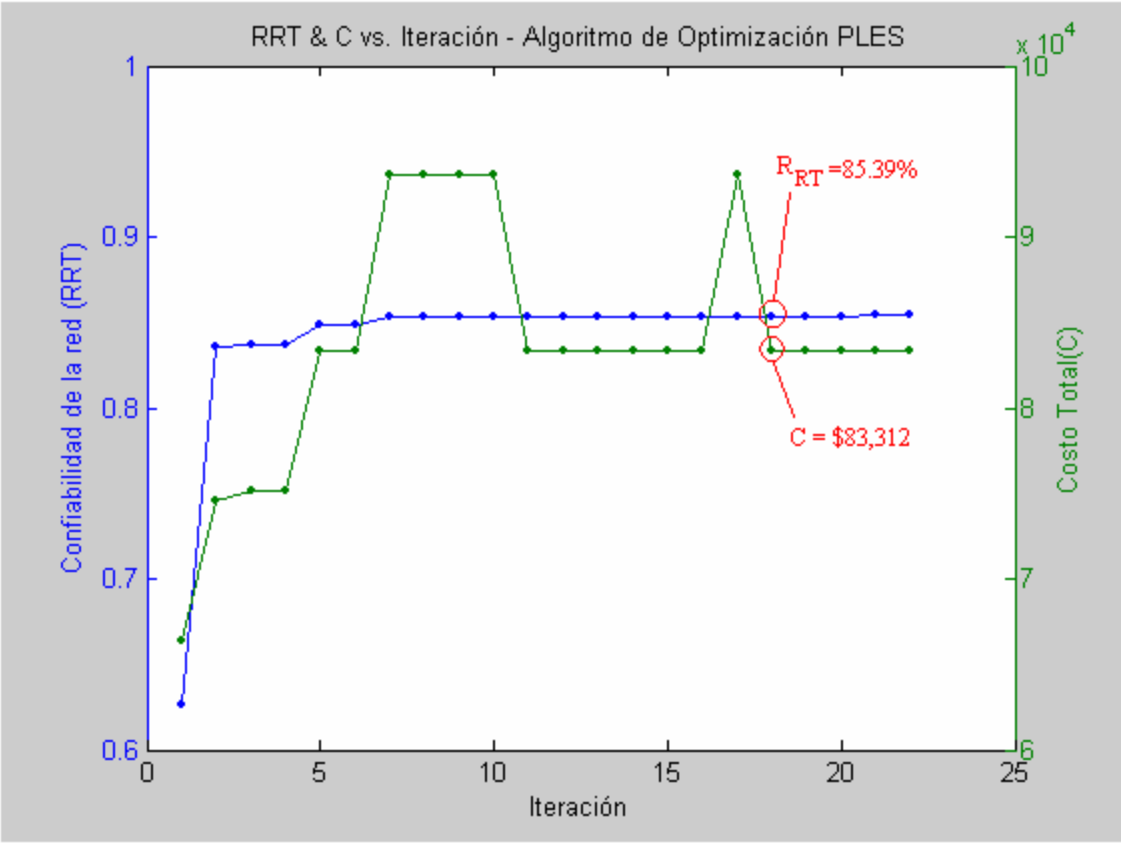


Figura 23. Confiabilidad de la red y costo asociado calculados en cada iteración del algoritmo

En la Figura 24 se presenta un diagrama de flujo que resume el procedimiento seguido por el algoritmo propuesto, PLES, en búsqueda de la solución al problema.

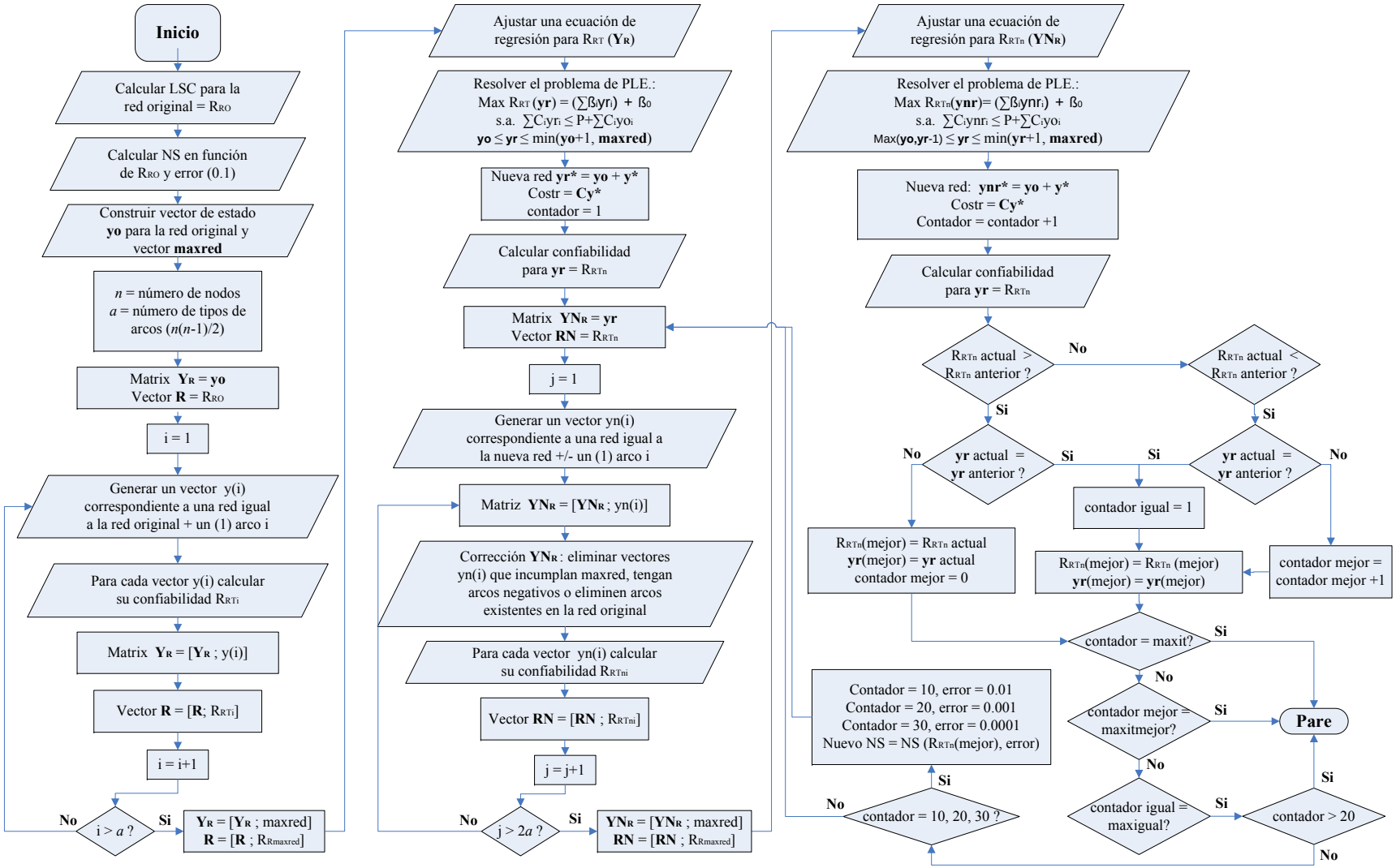


Figura 24. Diagrama de flujo para el algoritmo propuesto utilizando programación lineal entera secuencial

4 Análisis y Resultados

Una vez definida la metodología para el cálculo aproximado de las confiabilidades de las redes complejas mediante simulación de Monte Carlo (SMC) y descritos los dos heurísticos de optimización, Algoritmos Genéticos (AG) y Programación Lineal Entera Secuencial (PLES), se procedió a desarrollar dos experimentos para comparar los heurísticos y las metodologías para calcular la confiabilidad de la red.

En este capítulo se describen los diseños de experimentos desarrollados y se presentan los resultados y análisis obtenidos.

4.1 Diseño de Experimentos

El objetivo de los experimentos factoriales que a continuación se describen es comparar la confiabilidad máxima alcanzada y el tiempo de ejecución de los dos heurísticos de optimización, AG vs. PLES, y las dos metodologías para calcular la confiabilidad de la red, SMCRM vs. SMCAD.

Dos métodos para calcular la confiabilidad de la red empleando SMC fueron planteados: 1) SMCRM, empleando rutas mínimas, discutido en la sección 3.1.2.1; y 2) SMCAD, aplicando el algoritmo de Dijkstra, presentado en la sección 3.1.2.2.

En el capítulo anterior se concluyó de los resultados preliminares que las redes a evaluar tendrán entre 5 y 20 nodos; la confiabilidad de las redes con menos de 12 nodos serán aproximadas empleando SMCRM y SMCAD, mientras que la confiabilidad de las redes con más de 12 nodos, dado que la metodología de SMCRM demanda mucho tiempo computacional para estimar la confiabilidad de redes de este tamaño, sólo serán aproximadas con SMCAD. La confiabilidad de los arcos de las redes están entre 0 y 0.3 ($0 < r_p < 0.3$).

De acuerdo a estas conclusiones, se desarrollan los dos experimentos factoriales completos descritos a continuación:

Primer Experimento:

Factores:

1. Tamaño de la red: 6, 9 y 12 nodos.
2. Algoritmos: AG y PLES.

3. Métodos para calcular la confiabilidad: SMCRM y SMCAD.

Variables de respuesta:

1. Valor de la función objetivo: confiabilidad óptima de la red con arcos redundantes (R_{RT})
2. Tiempo de ejecución (t_e)

Segundo Experimento: Cálculo de la confiabilidad de las redes empleando SMCAD

Factores:

1. Tamaño de la red: 6, 9, 12, 16 y 20 nodos.
2. Algoritmos: AG y PLES.

Variables de respuesta:

1. Valor de la función objetivo: confiabilidad óptima de la red con arcos redundantes (R_{RT})
2. Tiempo de ejecución (t_e)

Para cada condición experimental se realizaron 5 réplicas considerando para cada tamaño de red (n) igual vector de estado para la red original (**yo**) y modificando en cada réplica el vector de costo (**C**), el vector de confiabilidad de los arcos de la red original (**ro**), el vector de confiabilidad de los arcos redundantes (**rr**) y el presupuesto de inversión (P). Los valores de los tres vectores fueron generados aleatoriamente siguiendo una distribución uniforme: para las confiabilidades entre 0 y 0.3 y para los costos entre $10,000 y $30,000. El presupuesto se determinó empleando la siguiente ecuación:

$$P = 0.8\sum_{i=1}^{a} C_i(maxred_i - yo_i) \tag{4.1}$$

Los vectores de estado (**yo**) de las redes utilizadas para los experimentos fueron generados de las redes tomadas de Kuo, Lu y Yeh (1999), sus diagramas de red se presentan en la Figura 25.

En la página 63 se presentan las gráficas de dispersión de las variables de respuesta, confiabilidad de la red, R_{RT} (Figura 26), y tiempo de ejecución, t_e (Figura 27), por algoritmo de optimización y método para calcular la confiabilidad. En la Figura 26 se observa que el crecimiento de los valores de R_{RT} a medida que n aumenta es lineal, sin embargo, la Figura 27 sugiere un comportamiento no lineal de t_e en n, por ello se propone una transformación logarítmica de t_e y se presenta en la Figura 28 las gráficas para el $\ln(t_e)$ confirmando una tendencia lineal de crecimiento en n.

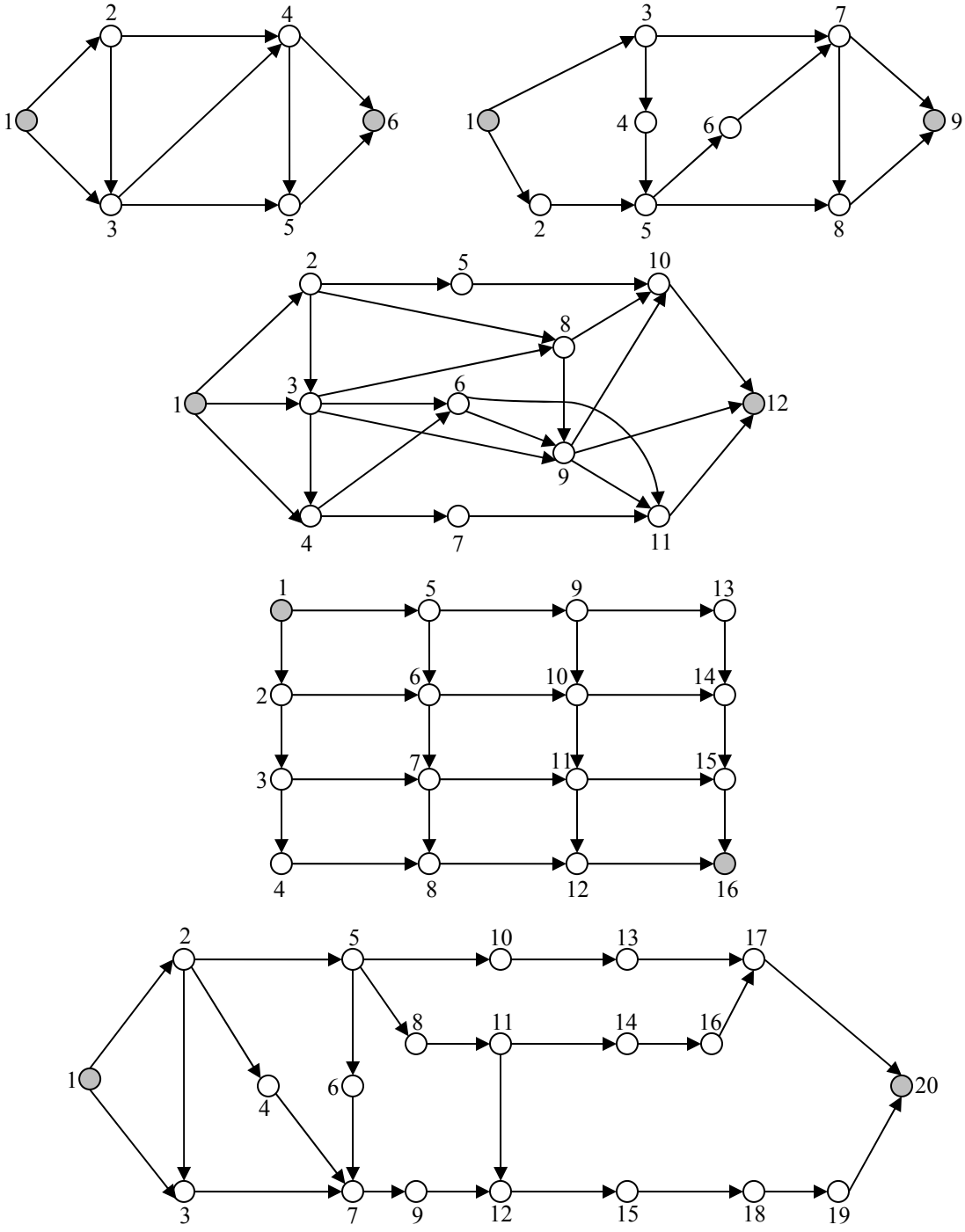


Figura 25. Redes complejas con 6, 9, 12, 16 y 20 nodos

La confiabilidad de la red, R_{RT}, pese a tener una tendencia lineal en n, como se explica en la sección 4.3., no cumple con la presunción de ANOVA de varianza constante por ser calculada como una proporción de éxitos en la SMC, por lo tanto, debe ser transformada para poder realizar el análisis estadístico.

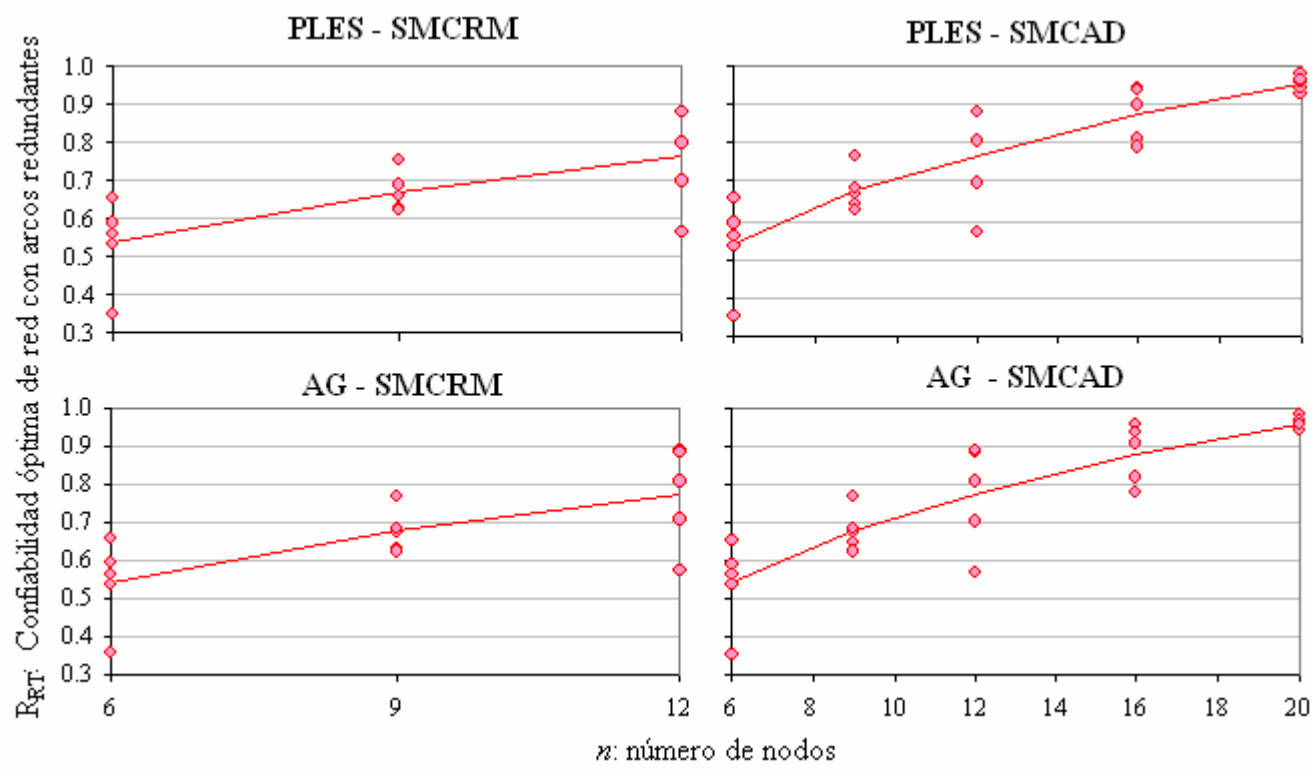


Figura 26. Gráficas de dispersión de la confiabilidad óptima de la red – R_{RT} vs. n

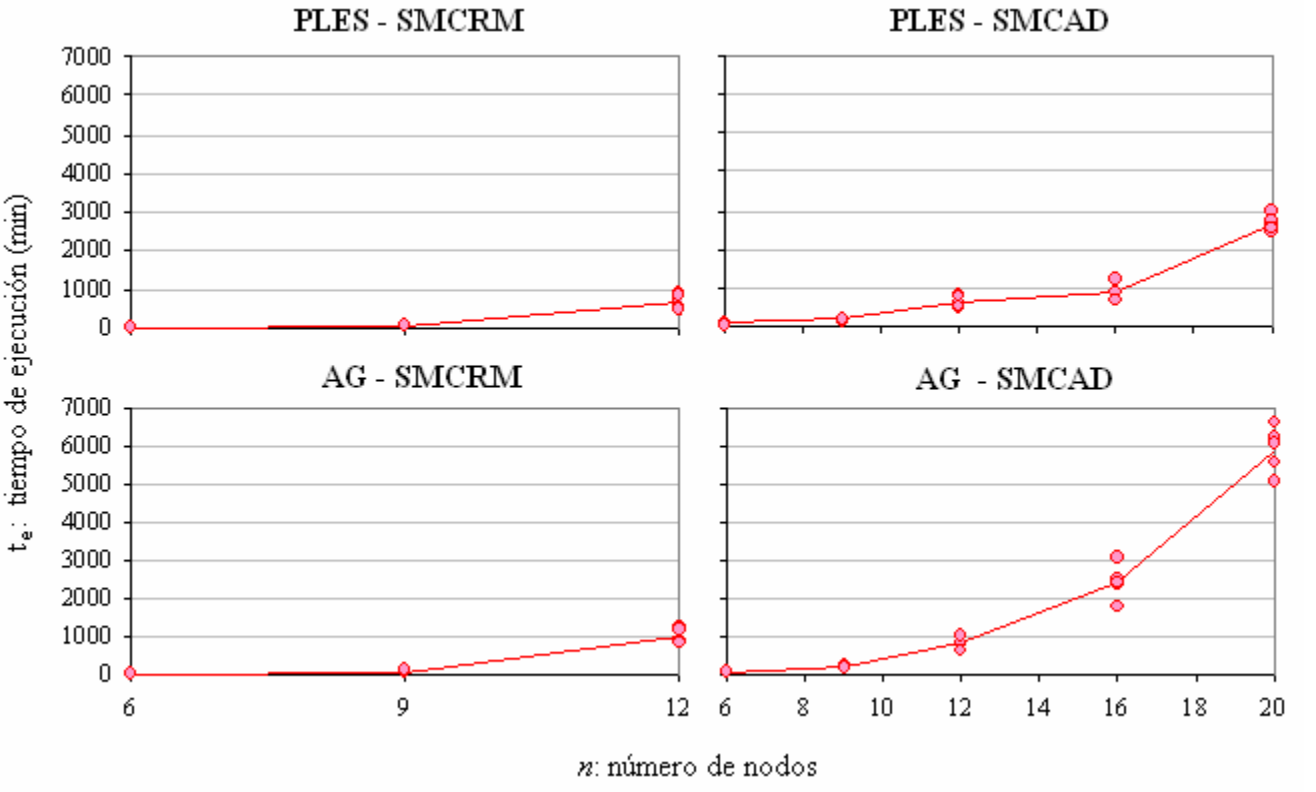


Figura 27. Gráficas de dispersión del tiempo de ejecución - t_e vs. n

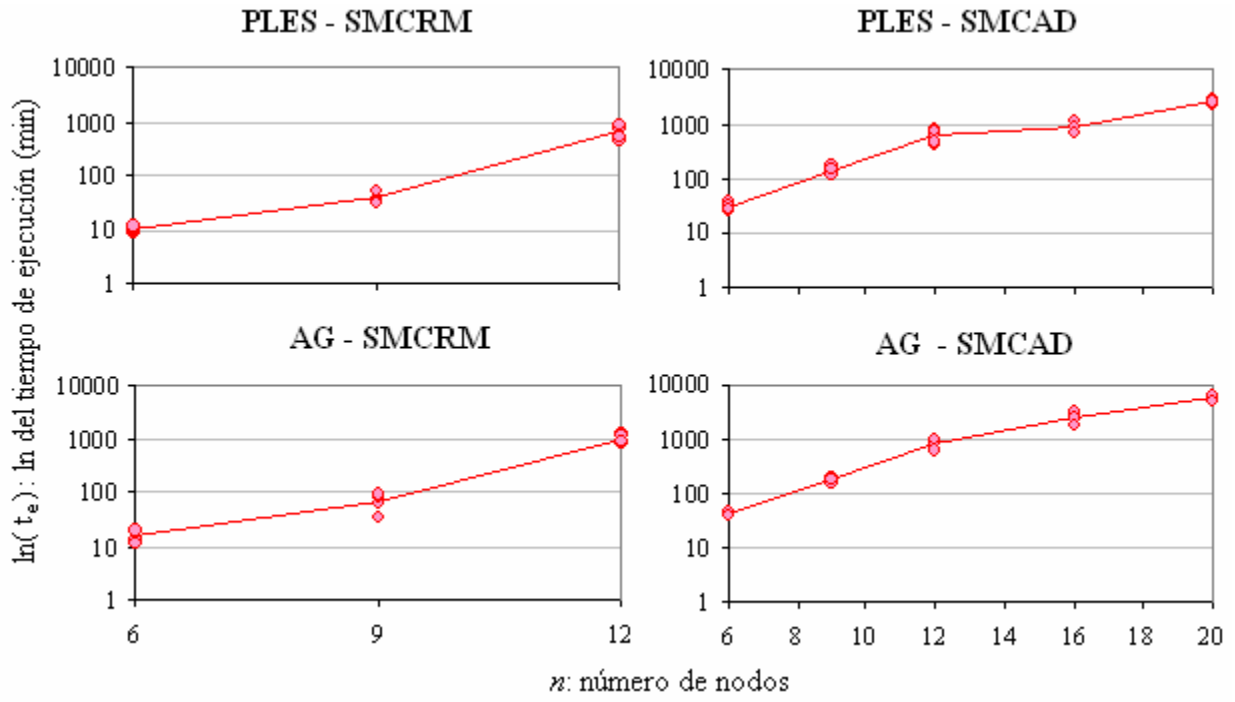


Figura 28. Gráficas de dispersión del logaritmo del tiempo de ejecución – ln(t_e) vs. n

Se plantean las Ecuaciones (4.2) y (4.3) que presentan los modelos de regresión lineal en forma general que van a ser ajustados para las variables de respuesta transformadas en función de los factores considerados en los experimentos.

Notación:

R'_{RT}: confiabilidad óptima de la red con arcos redundantes transformada

t_e: tiempo de ejecución transformado

n: número de nodos de la red

A: algoritmo de optimización

AG: Algoritmo Genético

PLES: Algoritmo de Programación Lineal Entera Secuencial

MC: método para calcular la confiabilidad

SMCRM: Simulación de Monte Carlo empleando Rutas Mínimas

SMCAD: Simulación de Monte Carlo empleando el Algoritmo de Dijkstra

$$R'_{RT} = \beta_0 + \beta_1 n + \begin{cases} \beta_2, \text{ si } A\text{=AG} \\ 0, \text{ si } A\text{=PLES} \end{cases} + \begin{cases} \beta_3, \text{ si } MC\text{=SMCRM} \\ 0, \text{ si } MC\text{=SMCAD} \end{cases} + \begin{cases} \beta_4, \text{ si } A\text{=AG, } MC\text{=SMCRM} \\ 0, \text{ de lo contrario} \end{cases} \quad (4.2)$$

$$\ln(t_e) = \beta_0 + \beta_1 n + \begin{cases} \beta_2, \text{ si } A\text{=AG} \\ 0, \text{ si } A\text{=PLES} \end{cases} + \begin{cases} \beta_3, \text{ si } MC\text{=SMCRM} \\ 0, \text{ si } MC\text{=SMCAD} \end{cases} + \begin{cases} \beta_4, \text{ si } A\text{=AG, } MC\text{=SMCRM} \\ 0, \text{ de lo contrario} \end{cases} \quad (4.3)$$

4.2 Análisis del tiempo de ejecución de los algoritmos

El tiempo de ejecución (t_e) es una de las variables de respuesta que se obtuvo de los dos experimentos con el fin de comparar el desempeño de los algoritmos de optimización heurística y los métodos para calcular la confiabilidad de la red. El tiempo de ejecución es el tiempo total exigido por el algoritmo programado en Matlab para encontrar la respuesta que maximice la confiabilidad de la red en estudio hasta que una de los condiciones de terminación se cumpla. Los algoritmos de optimización fueron ejecutados en una computadora Intel Pentium M, con procesador de 1.6 GHz y 512 MB de memoria RAM.

Para realizar el análisis de varianza, ANOVA, se consideró la variable transformada $\ln(t_e)$ con el fin de cumplir con las presunciones de ANOVA de normalidad de los residuales y varianza constante. Las gráficas de la Figura 28 confirman la relación lineal entre $\ln(t_e)$ y n.

La Tabla 27 presenta el resultado de ANOVA para los datos del primer experimento concluyendo, con un nivel de significancia del 5% que el tamaño de la red (n), el tipo de algoritmo de optimización (A) y la metodología para calcular la confiabilidad (MC) influyen en el tiempo de ejecución del algoritmo (t_e); la interacción de los factores A y MC no es significativa, por lo tanto, en la Tabla 28 se presenta el ANOVA para los datos del primer experimento incluyendo únicamente los factores significativos. De manera consistente, para el ANOVA obtenido para los datos del segundo experimento, presentado en la Tabla 29, se concluye con un nivel de significancia del 5% que el tamaño de red (n) y el tipo de algoritmo de optimización (A) influyen en el tiempo de ejecución del algoritmo (t_e).

Las ecuaciones (4.4) y (4.5) presentan los modelos de regresión para el $\ln(t_e)$ del primer y segundo experimento con R^2 de 93.7% y 93.6%, respectivamente.

$$\begin{aligned} \ln(t_e) &= -0.6302 + 0.5946n + \begin{cases} 0.1898, \text{ si } A = \text{AG} \\ 0, \text{ si } A = \text{PLES} \end{cases} + \begin{cases} -0.3301, \text{ si } MC = \text{SMCRM} \\ 0, \text{ si } MC = \text{SMCAD} \end{cases} \\ t_e &= \exp\left(-0.6302 + 0.5946n + \begin{cases} 0.1898, \text{ si } A = \text{AG} \\ 0, \text{ si } A = \text{PLES} \end{cases} + \begin{cases} -0.3301, \text{ si } MC = \text{SMCRM} \\ 0, \text{ si } MC = \text{SMCAD} \end{cases} \right) \end{aligned} \quad (4.4)$$

Tabla 27. ANOVA para el ln(t_e) del primer experimento
(incluye todos los factores e interacciones)

Factor	Grados de Libertad	Suma de Cuadrados	Mínimos Cuadrados	Estadístico F	Valor p
Tamaño de la Red (*n*)	1	127.271	127.271	824.72	0.000
Algoritmo (*A*)	1	2.163	2.163	14.02	0.000
Método de Confiabilidad (*MC*)	1	6.540	6.540	42.38	0.000
A * *MC*	1	0.136	0.136	0.88	0.351
Error	55	8.488	0.154	R^2 =	94.13
Total	59	144.598		R^2 ajustado =	93.70

Tabla 28. ANOVA para el ln(t_e) del primer experimento
(incluye sólo factores significativos)

Factor	Grados de Libertad	Suma de Cuadrados	Mínimos Cuadrados	Estadístico F	Valor p
Tamaño de la Red (*n*)	1	127.271	127.271	826.43	0.000
Algoritmo (*A*)	1	2.163	2.163	14.05	0.000
Método de Confiabilidad (*MC*)	1	6.540	6.540	42.46	0.000
Error	56	8.488	0.154	R^2 =	94.04
Total	59	144.598		R^2 ajustado =	93.72

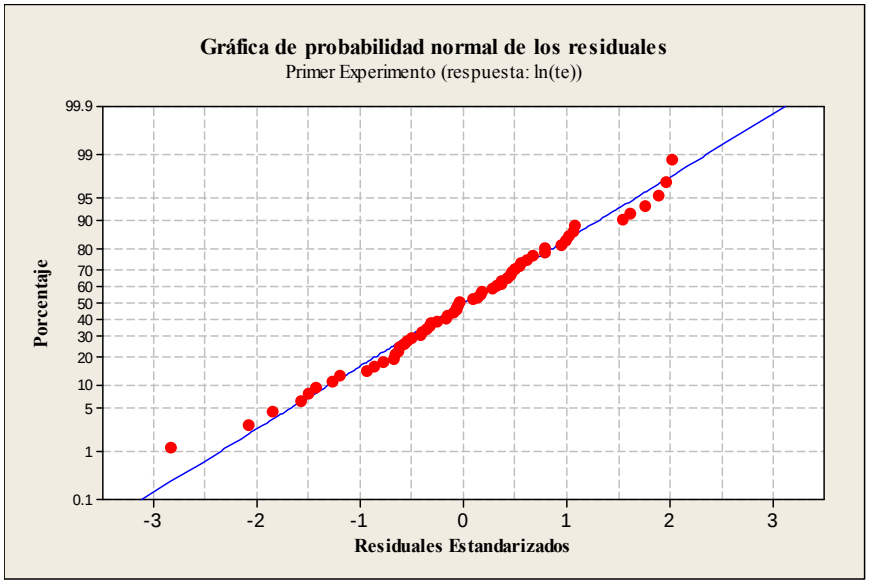


Figura 29. Gráfica de probabilidad normal de los residuales para el ln(t_e) - primer experimento

$$\ln(t_e) = 1.9974 + 0.3291n + \begin{cases} 0.2623, \text{ si } A = \text{AG} \\ 0, \text{ si } A = \text{PLES} \end{cases}$$
$$t_e = \exp\left(1.9974 + 0.3291n + \begin{cases} 0.2623, \text{ si } A = \text{AG} \\ 0, \text{ si } A = \text{PLES} \end{cases}\right) \quad (4.5)$$

Tabla 29. ANOVA para el $\ln(t_e)$ del segundo experimento

Factor	Grados de Libertad	Suma de Cuadrados	Mínimos Cuadrados	Estadístico F	Valor p
Tamaño de la Red (n)	1	133.413	133.413	703.67	0.000
Algoritmo (A)	1	3.440	3.440	18.14	0.000
Error	47	8.911	0.190	R^2 =	93.89
Total	49	145.765		R^2 ajustado =	93.63

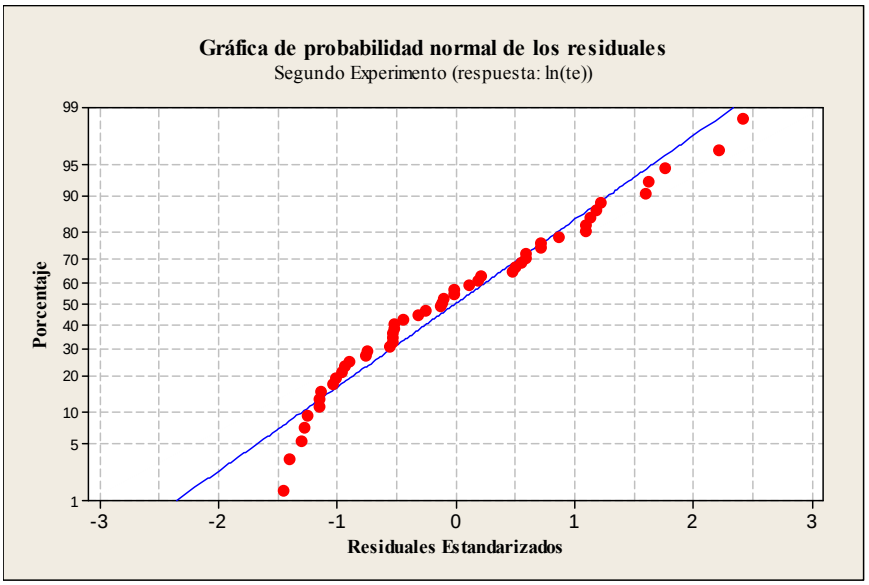


Figura 30. Gráfica de probabilidad normal de los residuales para el $\ln(t_e)$ - segundo experimento

Gráficamente se observa un comportamiento normal de los residuales para los dos modelos de regresión (Figura 29 y Figura 30), no obstante, se realizó una prueba de normalidad Anderson Darling para los residuales obteniendo un valor p de 0.831 para el primer experimento y 0.059 para el segundo experimento, concluyendo con un nivel de significancia del 5% que no hay suficiente información estadística para rechazar la hipótesis nula de normalidad.

Las figuras que se presentan a continuación muestran las gráficas de ln(t_e) predicho vs. n para los algoritmos de optimización en interacción con los métodos para calcular la confiabilidad de la red.

A partir de estas gráficas se puede concluir que:

- Para redes con menos de 12 nodos el menor tiempo de ejecución es el demandado por el algoritmo PLES calculando la confiabilidad con SMCRM (Figura 31).
- El método para calcular la confiabilidad de la red empleando las rutas mínimas en la Simulación de Monte Carlo (SMCRM) es eficiente en términos de tiempo para redes pequeñas con menos de 12 nodos. A partir de los 12 nodos, el método para calcular la confiabilidad de la red aplicando el algoritmo de Dijkstra (SMCAD) es más efectivo dado que estima más rápido los éxitos de la simulación (conexión entre nodo origen y nodo destino) como se había pronosticado en el capítulo anterior en la sección de Simulación de Monte Carlo.
- El heurístico de optimización propuesto en esta tesis basado en programación lineal entera secuencial, PLES, resulta ser para todos los tamaños de red (n) un algoritmo más rápido que el algoritmo genético, AG, encontrando en menor tiempo y menor número de iteraciones la respuesta óptima al problema de maximización de la confiabilidad de una red compleja adicionando arcos en redundancia. Esta diferencia de tiempos de ejecución de los algoritmos crece exponencialmente a medida que n aumenta.

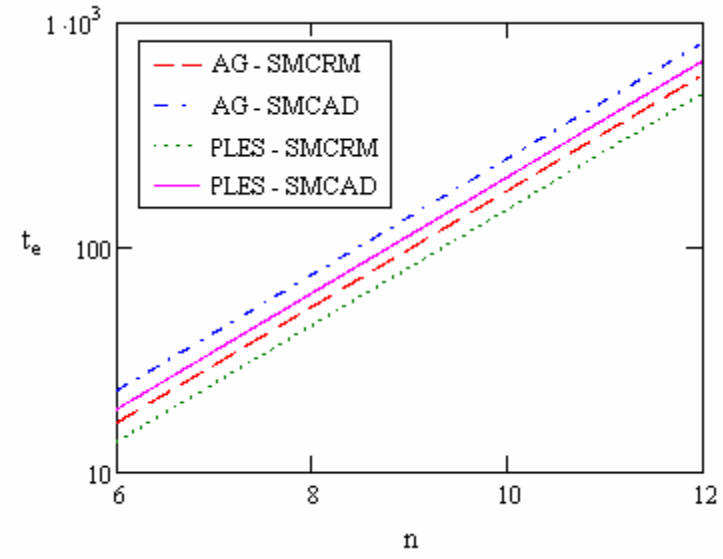


Figura 31. Gráfica de t_e predicho vs. n para los datos primer experimento

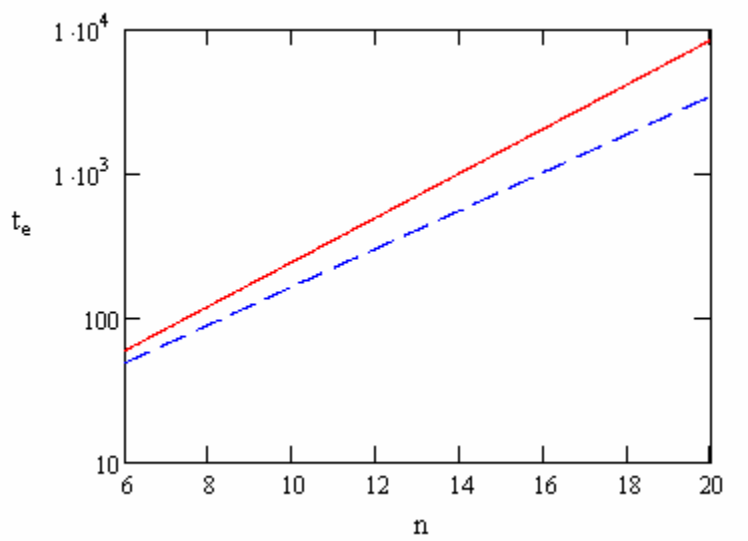


Figura 32. Gráfica de t_e predicho vs. n para los datos del segundo experimento

4.3 Análisis del valor de la función objetivo (R_{RT})

La confiabilidad de la red, como se explicó en el capítulo anterior, es estimada empleando Simulación de Monte Carlo, se realizan *NS* corridas de simulación y la proporción de éxitos en *NS* corridas es el valor aproximado de la confiabilidad.

Las proporciones estimadas (R_{RT}) tienen una varianza que depende de la misma proporción y, por lo tanto, la presunción de ANOVA de varianza constante no se cumple.

Mandy (1976) propuso la siguiente transformación exponencial aplicable para proporciones:

$$R_{RT}' = \frac{\exp(\phi R_{RT}) - 1}{\phi} \tag{4.6}$$

El parámetro de la transformación, ø, típicamente se estima empleando el principio de máxima verosimilitud, que consiste en este caso en encontrar el valor del parámetro que maximiza la probabilidad de que los residuales del conjunto de datos sigan una distribución normal; sin embargo, en este trabajo se usó el método que se describe a continuación.

Se transformó la variable R_{RT} para valores de ø entre 0.1 y 4 ajustando una ecuación de regresión lineal a la variable transformada en función de los factores del experimento (algoritmo (*A*), método para calcular la confiabilidad (*MC*) y tamaño de red (*n*)), se realizaron tres pruebas de normalidad (Anderson Darling, A-D, Ryan Joiner, R-J, y Kolmogorov Smirnov, K-S) para los residuales calculados a partir de la regresión lineal y se almacenaron los valores p de cada prueba.

En la Tabla 30 se observa que, para los dos experimentos, con ø = 2.5 se obtiene el máximo valor p para las tres pruebas, por lo tanto, para la transformación exponencial con este valor de ø se concluye que no hay suficiente información estadística para rechazar la hipótesis nula que indica que los residuales siguen una distribución normal.

Tabla 30. Valor p para las tres pruebas de normalidad

ø	Valor p (primer experimento)			Valor p (segundo experimento)		
	A - D	**R - J**	**K - S**	**A - D**	**R - J**	**K – S**
0.1	< 0.005	< 0.01	< 0.01	0.029	0.031	< 0.01
0.5	< 0.005	< 0.01	0.041	0.063	0.054	0.043
1	< 0.005	< 0.01	0.103	0.196	> 0.1	> 0.15
2	0.050	0.043	> 0.15	0.601	> 0.1	> 0.15
2.5	0.066	0.049	> 0.15	0.719	> 0.1	> 0.15
3	0.060	0.049	> 0.15	0.675	> 0.1	> 0.15
4	0.029	0.043	0.069	0.417	> 0.1	> 0.15

Con los resultados del análisis anterior, se aplica la transformación exponencial presentada en la Ecuación (4.6), con parámetro ø = 2.5, a la variable de respuesta R_{RT} de los dos experimentos.

La Tabla 31 presenta el resultado de ANOVA para los datos del primer experimento concluyendo con un nivel de significancia del 5% que únicamente el factor tamaño de la red (n) influye en la confiabilidad de la red con redundancia (R_{RT}). Esta misma conclusión es la obtenida a partir de los resultados de ANOVA para el segundo experimento, presentados en la Tabla 33.

Las ecuaciones (4.7) y (4.8) presentan los modelos de regresión obtenidos para la variable transformada de la R_{RT} para el primer y segundo experimento, respectivamente.

$$\frac{\exp(\phi R_{RT})-1}{\phi} = -0.1033 + 0.2124n$$
$$R_{RT} = \frac{\ln(\phi(-0.1033 + 0.2124n)+1)}{\phi} \qquad (4.7)$$

Tabla 31. ANOVA para la confiabilidad óptima transformada (R_{RT} ') - primer experimento (incluye todos los factores e interacciones)

Factor	Grados de Libertad	Suma de Cuadrados	Mínimos Cuadrados	Estadístico F	Valor p
Tamaño de la Red (n)	1	16.2476	16.2476	54.08	0.000
Algoritmo (A)	1	0.0017	0.0017	0.01	0.941
Método de Confiabilidad (MC)	1	0.0012	0.0012	0.00	0.951
A * MC	1	0.0020	0.0020	0.01	0.935
Error	55	16.5246	0.3004	R^2 =	49.58
Total	59	32.7770		R^2 ajustado =	45.92

Tabla 32. ANOVA para la confiabilidad óptima transformada (R_{RT} ') – primer experimento (incluye sólo factores significativos)

Factor	Grados de Libertad	Suma de Cuadrados	Mínimos Cuadrados	Estadístico F	Valor p
Tamaño de la Red (n)	1	16.2476	16.2476	54.08	0.000
Error	58	16.5246	0.3004	R^2 =	49.58
Total	59	32.7770		R^2 ajustado =	45.92

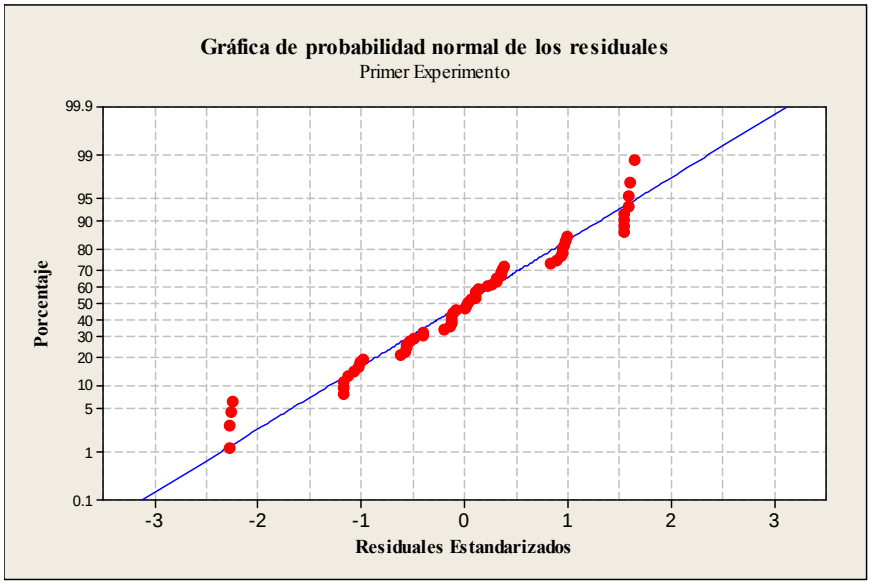


Figura 33. Gráfica de probabilidad normal de los residuales para la R_{RT}'- primer experimento

$$\frac{\exp(\phi R_{RT})-1}{\phi} = 0.0093 + 0.2001n$$
$$R_{RT} = \frac{\ln(\phi(0.0093+0.2001n)+1)}{\phi} \quad (4.8)$$

Tabla 33. ANOVA para la confiabilidad óptima transformada (R_{RT} ') - segundo experimento (incluye todos los factores e interacciones)

Factor	Grados de Libertad	Suma de Cuadrados	Mínimos Cuadrados	Estadístico F	Valor p
Tamaño de la Red (n)	1	49.300	49.300	190.17	0.000
Algoritmo (A)	1	0.000	0.000	0.000	0.999
Error	47	12.184	0.259	R^2 =	80.13
Total	49	61.484		R^2 ajustado =	79.34

Tabla 34. ANOVA para la confiabilidad óptima transformada (R_{RT} ') - segundo experimento (incluye sólo factores significativos)

Factor	Grados de Libertad	Suma de Cuadrados	Mínimos Cuadrados	Estadístico F	Valor p
Tamaño de la Red (n)	1	49.300	49.300	190.17	0.000
Error	48	12.184	0.259	R^2 =	80.13
Total	49	61.484		R^2 ajustado =	79.34

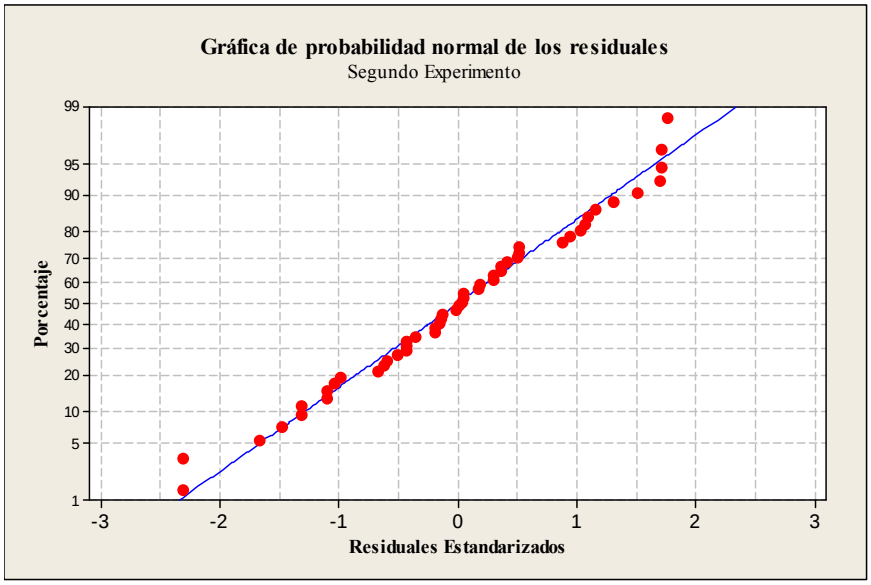


Figura 34. Gráfica de probabilidad normal de los residuales para la R_{RT}'- segundo experimento

A partir de los resultados de los análisis de varianza se concluye que el heurístico de optimización PLES propuesto en esta investigación maximiza la confiabilidad de la red con igual eficiencia que el AG planteado. Dado que el único factor significativo en la confiabilidad de la red con redundancia (R_{RT}) es el tamaño de la red (n), se unen los datos de los dos experimentos

obteniendo un solo modelo de regresión para R_{RT} en función de n presentado en la Ecuación (4.9).

$$\frac{\exp(\phi R_{RT})-1}{\phi} = -0.01 + 0.202n$$
$$R_{RT} = \frac{\ln(\phi(-0.01+0.202n)+1)}{\phi} \tag{4.9}$$

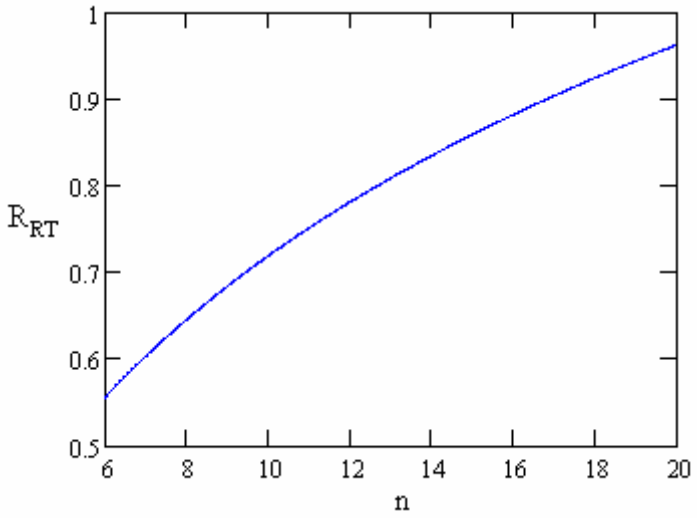


Figura 35. Gráfica de R_{RT} predicho vs. n

La confiabilidad de la red es la probabilidad de que esta funcione satisfactoriamente durante un período de tiempo, para el caso de esta investigación la red funciona si existe conexión entre los nodos origen y final. En la Figura 35 se observa que la confiabilidad total de la red es creciente cuando el tamaño de la red aumenta, esto se debe a que a medida que la red se hace grande existen más rutas para llegar desde el nodo origen hasta el nodo final y por lo tanto la probabilidad de éxito (confiabilidad) es mayor.

En el 15% de los experimentos realizados se obtuvo la misma solución al problema de optimización mediante los dos algoritmos estudiados; el AG obtuvo en el 45% de los experimentos un mayor valor en la función objetivo que PLES; PLES obtuvo en el 40% de los experimentos un mayor valor en la función objetivo que el AG. Estos resultados indican que no hay una tendencia de superioridad de un algoritmo sobre otro en términos de la respuesta que

generan al problema de optimización de redes estudiado. La figura que se presenta a continuación confirma gráficamente los porcentajes listados:

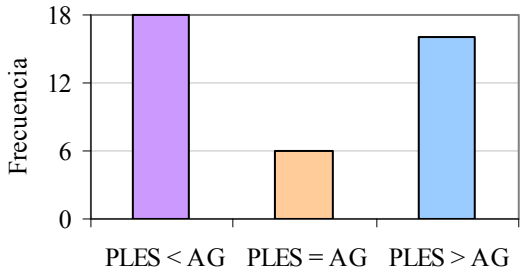


Figura 36. Comparación de soluciones obtenidas con PLES y AG

5 Conclusiones e Investigación Futura

5.1 Conclusiones

En este trabajo de tesis dos algoritmos de optimización heurística son usados y comparados para determinar el número y la ubicación óptima de los arcos adicionales que incrementan la confiabilidad de una red compleja con flujo desde el nodo de menor índice hacia el nodo de mayor índice: un algoritmo genético (AG) y un algoritmo basado en programación lineal entera secuencial (PLES).

Considerando que el cálculo exacto de la confiabilidad para redes complejas con todos los pares de arcos conectados es computacionalmente exigente, se estimó la confiabilidad aproximada de las redes empleando simulación de Monte Carlo (SMC). Dos metodologías de SMC fueron planteadas y evaluadas, SMCRM que utiliza las rutas mínimas de la red para determinar los éxitos de la simulación y SMCAD que aplica el algoritmo de Dijkstra.

Se desarrollaron dos experimentos factoriales con el fin de comparar la confiabilidad máxima alcanzada (R_{RT}) y el tiempo de ejecución (t_e) de los dos heurísticos de optimización, AG vs. PLES, y las dos metodologías para calcular la confiabilidad de la red, SMCRM vs. SMCAD. A partir de estos experimentos se concluyó que la confiabilidad máxima alcanzada sólo se veía afectada por el tamaño de la red (n) y por lo tanto los dos heurísticos, PLES y AG, maximizan la confiabilidad de la red (R_{RT}) con igual eficiencia. Adicionalmente, los resultados de los experimentos también permitieron concluir que el heurístico propuesto PLES resultó ser un algoritmo más rápido que AG para todos los tamaños de red.

Otras conclusiones relevantes se presentan a continuación:

- A partir de los experimentos desarrollados para estimar los parámetros de las redes a evaluar, se concluye con un nivel de significancia del 5% que la precisión del estimado de confiabilidad de la SMC (error = confiabilidad simulada – confiabilidad exacta) se ve afectado por el tamaño de la red (n), la confiabilidad promedio asignada a los arcos (r_p), y el número de corridas de simulación (NS).
- Evaluando los tiempos de simulación (t_s) y los resultados de los experimentos, se determina que la metodología de SMCRM es eficiente en tiempo de computadora para el cálculo de confiabilidad de redes con menos de 12 nodos. Para las redes con 12 o más

nodos, el método de SMCAD es más rápido en el cálculo de la confiabilidad de las redes que la SMCRM.

- Se descarta trabajar con redes con arcos con confiabilidades mayores que 0.5 dado que, en estos casos, la confiabilidad de la red es cercana a 1 y la oportunidad de mejorarlas mediante adición de arcos es limitada. Dado que el |error| es máximo para valores de r_p alrededor de 0.3, se decidió que la confiabilidad promedio de los arcos de las redes a incluir en el estudio sea menor a 0.3 ($r_p < 0.3$).

Se destaca como principal contribución de esta investigación el desarrollo y programación del heurístico de optimización basado en programación lineal entera secuencial, PLES, algoritmo que resultó ser considerablemente más rápido en encontrar la solución de asignación de redundancia que maximiza la confiabilidad de la red en comparación con el Algoritmo Genético evaluado.

La metodología propuesta en ésta tesis considera la opción de adicionar todos los posibles tipos de arcos de la red ($a= n\ (n\text{-}1)/2$) incluyendo aquellos que conectan nodos que en la red original no se encontraban conectados.

PLES encontró soluciones para redes con máximo 20 nodos y 190 posibles tipos de arcos por limitantes de recursos de computadora, no obstante el algoritmo esta en capacidad de solucionar problemas de optimización de la confiabilidad de redes con un número de nodos superior a 20 empleando computadoras con mayor capacidad.

5.2 Trabajo Futuro

A partir del desarrollo de esta tesis se detectaron los temas que se describen a continuación como posibles trabajos futuros en la línea de investigación:

- Continuando con la metodología propuesta, considerar el mismo problema de optimización de redes complejas sin incluir la limitante de redes dirigidas con flujo unidireccional.
- Evaluar el problema de optimización de la confiabilidad de redes complejas considerando no sólo fallas en los arcos sino también fallas en los nodos.

- Plantear el problema para redes con múltiples destinos, donde se requiera evaluar la conexión entre el nodo origen con los n-1 nodos restantes de la red para estimar su confiabilidad.
- Resolver el problema para redes con más de 20 nodos empleando mejores recursos de computadora y analizando alternativas más rápidas que las simulaciones de Monte Carlo propuestas para estimar la confiabilidad aproximada de las redes.
- Emplear la metodología del algoritmo de Programación Lineal Entera Secuencial, PLES, para desarrollar un algoritmo de optimización heurística explorando funciones de optimización no lineal (p.e. función cuadrática).

Referencias

Aneja, Y., R. Chandrasekaran y K. Fair (2004) "Minimal-cost system reliability with discrete-choice sets for components". *IEEE Transactions on Reliabilit*y, Vol.53, No.1: 71-76.

Billinton, R. y R. Allan (1994) "Reliability Evaluation of Engineering Systems: Concepts and Techniques". Plenum Press.

Chao, M. y G. Lin (1984) "Economical design of large consecutive-k-out-of-n : F system". *IEEE Transactions on Reliability*, R-33: 411-3.

Coit, D. y A. Smith (1995) "Solving the redundancy allocation problem using a combined neural network / genetic algorithm approach". *Computers & Operations Research*.

Coit, D. y A. Smith (1996) "Reliability Optimization of Series-Parallel Systems using a Genetic Algorithm". *IEEE Transaction on Reliability*, Vol. 45, No.2: 254-260.

Dengiz, B., F. Antiparmak y A. Smith (1997) "Local search genetic algorithm for optimal design of reliable networks". *IEEE Transactions on Evolutionary Computation*, Vol.1, No.3: 179-188.

Dijkstra, Edsger (1959) "A note on two problems in connection with graphs". *Numerische Mathematik*, Vol. 1, No.1: 269-271.

Ebeling, C. (2005) "An Introduction to Reliability and Maintainability Engineering". Waveland Press, Inc.

Fotuhi, M., R. Villinton, T.S.Munian y B. Vinayagam (2004) "A Novel Approach to Determine Minimal Tie-Sets of Complex Network". *IEEE Transactions on Reliability*, Vol.53, No.1: 61-70.

Jacobson, D. y S. Arora (1996) " Simultaneous allocation of reliability & redundancy using simplex search". *IEEE Annual Reliability and Maintainability Symposium*. 243-250.

Kim, J. y B. Yum (1993) "A Heuristic method for solving redundancy optimization problems in complex systems". *IEEE Transactions on Reliability*, Vol.42, No4: 572-578.

Kohda, T. y K. Inoue (1982) "A reliability optimization method for complex systems with the criterion of local optimality". *IEEE Transactions on Reliability*, Vol. R-31: 109-111.

Kuo, W. y W. Zuo (2003) "Optimal Reliability Modeling. Principles and Applications". John Wiley & Sons, Inc.

Kuo, W. y V.R. Prasad (2000) "An annotated overview of system reliability optimization". *IEEE Transactions on Reliability*, R-49(2):176-178.

Kuo, S., S. Lu y F. Yeh (1999). "Determining terminal pair reliability based on edge expansion diagrams using OBDD". *IEEE Transactions on Reliability*, 48(3), 234–246.

Li, D. y Y. Haimes (1992). "A decomposition method for optimization of large system reliability". IEEE Transactions on Reliability, Vol.41: 183 – 188.

Mandy, B.F.J. (1976). "Exponential data transformations". *Statistician,* 36, 55-57.

Mitchell, Melanie (1998). "An Introduction to Genetic Algorithms". The MIT Press, Massachusetts.

Montgomery, D. y E.A. Peck (1992) "Introduction to linear regression analysis". John Wiley & Sons, Inc.

Ravi, V., B.S.N. Murty y P.J. Reddy (1997) "Nonequilibrium simulated annealing algorithm applied to reliability optimization of complex systems". *IEEE Transactions on Reliability*, Vol.46, No.2: 233-239.

Taha, Hamdy (2006). "Operations Research: An Introduction" (8 Edition). Prentice Hall.

Ramírez-Marquez, J. y D. Coit (2005) "A Monte-Carlo simulation approach for approximating multi-state two-terminal reliability". *Reliability Engineering & System Safety*, Vol.87, No.2: 253-259.

Prasad, V., Y. Aneja y K. Fair (1991) "A heuristic approach to optimal assignment of components to a parallel-series network". *IEEE Transactions on Reliability*, Vol.40, No.5: 555-558.

Shi, D. (1987) "A new heuristic algorithm for constrained redundancy optimization in complex systems". *IEEE Transactions on Reliability*, Vol. R-36: 621-623.

Tillman, F., C. Hwang y W. Kuo (1977) "Optimization techniques for systems reliability with redundancy – a review". *IEEE Transactions on Reliability*, Vol.R-26: 148-4155.

Apéndice A. Programas (Matlab 7.0.1)

Algoritmo PLES: cálculo exacto de la confiabilidad de la red

```
%          Desarrollado y programado por: Paola A. Hernandez-Ramirez
%                   Planteado por:  Noel Artiles-Leon
%                                Mayo 2007
%     Heurístico basado en Programación Lineal Entera Secuencial (PLES)
%
% Función Objetivo:  Maximizar la confiabilidad de una red compleja
% mediante la adición de arcos redundantes con una restricción de
% presupuesto de inversión
%
% Evaluación exacta de la función objetivo empleando la función de
% estructura de la red.  Esta evaluación puede realizarse para redes con
% 7 nodos máximo.
%
% Variable de entrada:   Vector de máxima redundancia (maxred)
% Variables de salida: * Confiabilidad de la red original (Rso)
%                      * Confiabilidad de la red con arcos redundantes (Rsn)
%                      * Vector de arcos redundantes (xr)
%                      * Tiempo de ejecución del algoritmo (timetaken(seg))
%
% Debe prepararse un archivo .mat que contenga la información general de la
% red compleja a optimizar (Matriz de rutas mínimas (MP), vector de
% confiabilidades de los arcos de la red original (ro), vector de
% confiabilidades de los arcos redundantes (rr), costo de los arcos (cost),
% presupuesto (budget), función de estructura de la red (sft))

function [Rso,Rsn,xr,Cost,timetaken] = PLESexacta(maxred)

tic;
load DatosRed       % Archivo .mat con la información de la red
maxit=50;           % Número máx. de it. en búsqueda del óptimo
maxbest=10;         % Número máx. de it. después de encontrar un óptimo local
Rso=FER(ro,Az,sf,sft)   % Confiabilidad exacta de la red original
numa=size(ro,2);    %Número de arcos totales para la red
maxredv=maxred*ones(1,numa);    % Vector con número máximo de arcos
Xr=[zeros(1,numa);eye(numa);maxredv-x0];    % Matriz de estado inicial
for i=1:size(Xr,1)
    X(i,:)=Xr(i,:)+x0;
end
X1=[ones(size(X,1),1) X];  % Adición de la fila de unos para el intercepto
Y=zeros(size(X,1),1);  %Vector de confiabilidades para c/fila de X
Y(1)=Rso;
for i=2:size(X,1)
    rtemp=1-(1-ro).*(1-rr).^Xr(i,:);
    Y(i)=FER(rtemp,Az,sf,sft);
end
%Coeficientes (betas) de la ecuación de regresión lineal (Y=Bo+B1X1+B2X2..)
betas=regress(Y,X1);
betas1=betas(2:size(betas),1);
```

```
lb=zeros(1,numa);   % Límite inferior del vector solución
ub=ones(1,numa);    % Límite superior del vector solución
xint=1:numa;
%Solución inicial de PLE (x: vector de arcos redundantes que Max Rs)
[obj,x]=lp_solve(betas1',cost,budget,-1,lb,ub,xint);
%-------------------------------------------------------------------------%
xn=x0+x';                   % Nueva red (original + arcos redundantes)
xr=x;                       % Vector de arcos redundantes
rn=1-(1-ro).*(1-rr).^(x');  % Vector de confiabilidades para la nueva red
Rsn=FER(rn,Az,sf,sft);      % Confiabilidad exacta de la nueva red
Cost=cost*xr;               % Costo del vector de arcos redundantes
% Vectores para gráfica de iteración vs. función objetivo
conteo=1;
Rs=Rsn;
Costo=Cost;
% Vectores de verificación (todas las soluciones generadas sin reemplazo)
Rsver=Rsn;
xrver=xr;
Costver=Cost;
% Inicialización de variables para los criterios de parada
numop=0;
Rsntemp=0;
cont=1;
contbest=0;
check=0;
Rsnbest=Rsn;
xnbest=xn;
xrbest=xr;
rnbest=rn;
Costbest=Cost;
while check==0 && cont<maxit && contbest<maxbest
    % Construcción de la matriz de estado: Red solución inicial + un arco i
    % y Red solución inicial - un arco (i)
    Xr=[zeros(1,numa);eye(numa);-eye(numa);maxredv-xn];
    for i=1:size(Xr,1)
        X2(i,:)=Xr(i,:)+xn;
    end
    % Eliminando los vectores que contienen soluciones con arcos que
    % superan el máximo establecido de redundancia, soluciones con arcos
    % negativos y aquellas que eliminan arcos que existen en la red original
    del=zeros(size(X2,1),1);
    for i=1:size(X2,1)
        if isempty(find(X2(i,:)>maxred))==0 || isempty(find(X2(i,:)<0))==0
            del(i)=1;
        elseif isempty(find(X2(i,:)==0))==0
            ceros=find(X2(i,:)==0);
            if x0(ceros)>0
                del(i)=1;
            end
        end
    end
    Xtemp=zeros(size(X2,1)-sum(del),size(X2,2));
    Xtemp1=Xtemp;
    c=0;
    for i=1:size(del,1)
        if del(i)==0
```

```
        Xtemp(i-c,:)=X2(i,:);
        Xtemp1(i-c,:)=Xr(i,:);
    else
        c=c+1;
    end
end
X2=Xtemp;
Xr=Xtemp1;
for i=1:size(Xr,1)
    Xr(i,:)=xr'+Xr(i,:);
end
X3=[ones(size(X2,1),1) X2];
% Vector de confiabilidades para la nueva matriz de estado
Y1=zeros(size(X2,1),1);
Y1(1)=Rsn;
for i=2:size(X2,1)
    rtemp=1-(1-ro).*(1-rr).^Xr(i,:);
    Y1(i)=FER(rtemp,Az,sf,sft);
end
% Coeficientes (betas) de la ecuación de regresión lineal
betas2=regress(Y1,X3);
betas3=betas2(2:size(betas2),1);
% Los vectores de límite inferior y superior de las variables de
% decisión estan sujetos a que puede quitarse un arco sólo si se ha
% agregado al menos 1 y se puede añadir uno nuevo si no se ha alcanzado
% el máximo de redundancia establecido (maxred)
lb=-ones(1,numa);
xtemp=xn-x0;
for i=1:size(lb,2)
    if xtemp(i)==0
        lb(i)=0;
    end
end
ub=ones(1,numa);
for i=1:size(ub,2)
    if xn(i)==maxred
        ub(i)=0;
    end
end
xint=1:numa;
budgetdis=budget-Cost;  % Presupuesto disponible para invertir
% Nueva solución de PLE
[obj,x1]=lp_solve(betas3', cost, budgetdis, -1, lb, ub, xint);
xr=xr+x1;
xn=x0+xr';
rn=1-(1-ro).*(1-rr).^(xr');
Rsn=FER(rn,Az,sf,sft);;
Cost=cost*xr;
% Actualización de los vectores de verificación
Rsver=[Rsver Rsn];
xrver=[xrver xr];
Costver=[Costver Cost];
if Rsn>Rsnbest
    if xr==xrbest
        Rsn=Rsnbest;
        numop=numop+1
```

```
    elseif xrbest>=xr
        Rsn=Rsnbest;
        xn=xnbest;
        xr=xrbest;
        rn=rnbest;
        Cost=Costbest;
        numop=numop+1
    else
        contbest=0
        Rsnbest=Rsn
        xnbest=xn;
        xrbest=xr;
        rnbest=rn;
        Costbest=Cost;
        numop=0
    end
elseif Rsn<Rsnbest
    if xr==xrbest
        Rsn=Rsnbest;
        numop=numop+1;
    elseif xr>=xrbest
        Rsnbest=Rsn
        xnbest=xn;
        xrbest=xr;
        rnbest=rn;
        Costbest=Cost;
        numop=0
    else
        contbest=contbest+1
        numop=numop+1
        Rsn=Rsnbest;
        xn=xnbest;
        xr=xrbest;
        rn=rnbest;
        Cost=Costbest;
    end
elseif Rsn==Rsnbest
    if xr==xrbest
        numop=numop+1
    elseif Cost>Costbest
        Rsn=Rsnbest;
        xn=xnbest;
        xr=xrbest;
        rn=rnbest;
        Cost=Costbest;
    end
end
if cont==maxit || contbest==maxbest
    xn=xnbest;
    xr=xrbest;
    rn=rnbest;
    Rsn=Rsnbest;
    Cost=Costbest;
end
cont=cont+1
%Vectores para gráfica de iteración vs. función objetivo
```

```
    conteo=[conteo cont];
    Rs=[Rs Rsn];
    Costo=[Costo Cost];
    if numop>=5
        check=1
        xn=xnbest;
        xr=xrbest;
        rn=rnbest;
        Rsn=Rsnbest;
        Cost=Costbest;
    end
end
timetaken=toc;      % Tiempo de ejecución del algoritmo

%Gráfica de iteración vs. función objetivo (Confiabilidad) & Costo Total
[AX,H1,H2] = plotyy(conteo,Rs,conteo,Costo,'plot');
set(get(AX(1),'Ylabel'),'String','Confiabilidad de la red (RRT)')
set(get(AX(2),'Ylabel'),'String','Costo Total(C)')
xlabel('Iteración');
title('RRT & C vs. Iteración - Algoritmo de Optimización PLES');
set(H1,'LineStyle','-','Marker','.')
set(H2,'LineStyle','-','Marker','.')
hold on
```

Algoritmo PLES: cálculo estimado de la confiabilidad de la red empleando SMCRM

```
%          Desarrollado y programado por: Paola A. Hernandez-Ramirez
%                   Planteado por:  Noel Artiles-Leon
%                                Mayo 2007
%     Heurístico basado en Programación Lineal Entera Secuencial (PLES)
%
% Función Objetivo:  Maximizar la confiabilidad de una red compleja
% mediante la adición de arcos redundantes con una restricción de
% presupuesto de inversión
%
% Evaluación de la función objetivo aplicando Simulación de Monte Carlo
% empleando la metodología de rutas mínimas (SMCRM)
%
% Variable de entrada:   Vector de máxima redundancia (maxred)
% Variables de salida: * Confiabilidad de la red original (Rso)
%                      * Confiabilidad de la red con arcos redundantes (Rsn)
%                      * Vector de arcos redundantes (xr)
%                      * Tiempo de ejecución del algoritmo (timetaken(seg))
%
% Debe prepararse un archivo .mat que contenga la información general de la
% red compleja a optimizar (Matriz de rutas mínimas (MP), vector de
% confiabilidades de los arcos de la red original (ro), vector de
% confiabilidades de los arcos redundantes (rr), costo de los arcos (cost)
% y presupuesto (budget))

function [Rso,Rsn,xr,Cost,timetaken] = PLESsmcrm(maxred)

tic;
```

```
load DatosRed       % Archivo .mat con la información de la red
maxit=50;           % Número máx. de it. en búsqueda del óptimo
maxbest=10;         % Número máx. de it. después de encontrar un óptimo local
Rso=LSC(ro,sft);    % Límite superior de la confiabilidad - Red Original
% Cálculo de NS en función de Rso y el error esperado
NS=ceil(16*Rso*(1-Rso)*(0.1^(-2)));  % e inicial = 0.1
if NS<10000         % NS mínimo = 100.000 corridas de simulación
    NS=10000;
end
numa=size(ro,2);    % Número de arcos totales para la red
maxredv=maxred*ones(1,numa);    % Vector con número máximo de arcos
%-------------------------------------------------------------------------%
Xr=[zeros(1,numa);eye(numa);maxredv-x0];    % Matriz de estado inicial
for i=1:size(Xr,1)
    X(i,:)=Xr(i,:)+x0;
end
X1=[ones(size(X,1),1) X];  % Adición de la fila de unos para el intercepto
Y=zeros(size(X,1),1);      % Vector de confiabilidades para c/fila de X
Y(1)=Rso;
for i=2:size(X,1)
    rtemp=1-(1-ro).*(1-rr).^Xr(i,:);
    Y(i)=SMCRM(MP,NS,rtemp);
end
%Coeficientes (betas) de la ecuación de regresión lineal (Y=Bo+B1X1+B2X2..)
betas=regress(Y,X1);
betas1=betas(2:size(betas),1);
lb=x0;                  % Límite inferior del vector solución
ub=x0+ones(1,numa);     % Límite superior del vector solución
for i=1:numa
    if ub(i)>maxredv(i)
        ub(i)=maxredv(i);
    end
end
xint=1:numa;
%Solución inicial de PLE (x: vector de arcos redundantes que Max Rs)
costx0=cost*x0';        % Costo de la red original
budgett=budget+costx0;  % Presupuesto total adicionando el costo de x0
[obj,x]=lp_solve(betas1',cost,budgett,-1,lb,ub,xint);
%-------------------------------------------------------------------------%
xn=x';                      % Nueva red (original + arcos redundantes)
xr=x-x0';                   % Vector de arcos redundantes
rn=1-(1-ro).*(1-rr).^(xr'); % Vector de confiabilidades para la nueva red
Rsn=SMCRM(MP,NS,rn);        % Confiabilidad de la nueva red
Cost=cost*xr;               % Costo del vector de arcos redundantes
Costt=cost*xn';              % Costo total de la nueva red
% Vectores para gráfica de iteración vs. función objetivo
conteo=1;
Rs=Rsn;
Costo=Cost;
% Vectores de verificación (todas las soluciones generadas sin reemplazo)
Rsver=Rsn;
xrver=xr;
Costver=Cost;
% Inicialización de variables para los criterios de parada
numop=0;
Rsntemp=0;
```

```
cont=1;
contbest=0;
check=0;
o=10; p=20; q=30;  % Rangos para ajuste de NS (disminución del error)
Rsnbest=Rsn;
xnbest=xn;
xrbest=xr;
rnbest=rn;
Costbest=Cost;
while check==0 && cont<maxit && contbest<maxbest
    % Construcción de la matriz de estado: Red solución inicial + un arco i
    % y Red solución inicial - un arco (i)
    Xr=[zeros(1,numa);eye(numa);-eye(numa);maxredv-xn];
    for i=1:size(Xr,1)
        X2(i,:)=Xr(i,:)+xn;
    end
    % Eliminando los vectores que contienen soluciones con arcos que
    % superan el máximo establecido de redundancia, soluciones con arcos
    % negativos y aquellas que eliminan arcos que existen en la red original
    del=zeros(size(X2,1),1);
    for i=1:size(X2,1)
        if isempty(find(X2(i,:)>maxredv))==0 || isempty(find(X2(i,:)<0))==0
            del(i)=1;
        elseif isempty(find(X2(i,:)==0))==0
            ceros=find(X2(i,:)==0);
            if sum(x0(ceros))>0
                del(i)=1;
            end
        end
    end
    Xtemp=zeros(size(X2,1)-sum(del),size(X2,2));
    Xtemp1=Xtemp;
    c=0;
    for i=1:size(del,1)
        if del(i)==0
            Xtemp(i-c,:)=X2(i,:);
            Xtemp1(i-c,:)=Xr(i,:);
        else
            c=c+1;
        end
    end
    X2=Xtemp;
    Xr=Xtemp1;
    for i=1:size(Xr,1)
        Xr(i,:)=xr'+Xr(i,:);
    end
    X3=[ones(size(X2,1),1) X2];
    % Vector de confiabilidades para la nueva matriz de estado
    Y1=zeros(size(Xr,1),1);
    Y1(1)=Rsn;
    for i=2:size(Xr,1)
        rtemp=1-(1-ro).*(1-rr).^Xr(i,:);
        Y1(i)=SMCRM(MP,NS,rtemp);
    end
    % Coeficientes (betas) de la ecuación de regresión lineal
    betas2=regress(Y1,X3);
```

```
    betas3=betas2(2:size(betas2),1);
    % Los vectores de límite inferior y superior de las variables de
    % decisión estan sujetos a que puede quitarse un arco sólo si se ha
    % agregado al menos 1 y se puede añadir uno nuevo si no se ha alcanzado
    % el máximo de redundancia establecido (maxred)
    lb=xn-ones(1,numa);     % Límite inferior del vector solución
    for i=1:numa
        if lb(i)<x0(i)
            lb(i)=x0(i);
        end
    end
    ub=xn+ones(1,numa);     % Límite superior del vector solución
    for i=1:numa
        if ub(i)>maxredv(i)
            ub(i)=maxredv(i);
        end
    end
    xint=1:numa;
    % Nueva solución de PLE
    [obj,x1]=lp_solve(betas3', cost, budgett, -1, lb, ub, xint);
    xr=x1-x0';
    xn=x1';
    rn=1-(1-ro).*(1-rr).^(xr');
    Rsn=SMCRM(MP,NS,rn);
    Cost=cost*xr;
    Costt=cost*xn';
%-------------------------------------------------------------------------%
    % Actualización de los vectores de verificación
    Rsver=[Rsver Rsn];
    xrver=[xrver xr];
    Costver=[Costver Cost];
    if Rsn>Rsnbest
        if xr==xrbest
            Rsn=Rsnbest;
            numop=numop+1
        elseif xrbest>=xr
            Rsn=Rsnbest;
            xn=xnbest;
            xr=xrbest;
            rn=rnbest;
            Cost=Costbest;
            numop=numop+1
        else
            contbest=0
            Rsnbest=Rsn
            xnbest=xn;
            xrbest=xr;
            rnbest=rn;
            Costbest=Cost;
            numop=0
        end
    elseif Rsn<Rsnbest
        if xr==xrbest
            Rsn=Rsnbest;
            numop=numop+1;
        elseif xr>=xrbest
```

```
        Rsnbest=Rsn
        xnbest=xn;
        xrbest=xr;
        rnbest=rn;
        Costbest=Cost;
        numop=0
    else
        contbest=contbest+1
        numop=numop+1
        Rsn=Rsnbest;
        xn=xnbest;
        xr=xrbest;
        rn=rnbest;
        Cost=Costbest;
    end
elseif Rsn==Rsnbest
    if xr==xrbest
        numop=numop+1
    elseif Cost>Costbest
        Rsn=Rsnbest;
        xn=xnbest;
        xr=xrbest;
        rn=rnbest;
        Cost=Costbest;
    end
end
if cont==maxit || contbest==maxbest
    xn=xnbest;
    xr=xrbest;
    rn=rnbest;
    Rsn=Rsnbest;
    Cost=Costbest;
end
cont=cont+1
if cont==o
    NS=min(50000,ceil(16*Rsnbest*(1-Rsnbest)*(0.01^(-2)))); % e = 0.01
    Rsnbest=SMCRM(MP,NS,rnbest);
    contbest=0;
    numop=0;
elseif cont==p
    NS=min(100000,ceil(16*Rsnbest*(1-Rsnbest)*(0.001^(-2)))); % e = 0.001
    Rsnbest=SMCRM(MP,NS,rnbest);
elseif cont==q
    NS=min(500000,ceil(16*Rsnbest*(1-Rsnbest)*(0.0001^(-2))));%e = 0.0001
    Rsnbest=SMCRM(MP,NS,rnbest);
end
%Vectores para gráfica de iteración vs. función objetivo
conteo=[conteo cont];
Rs=[Rs Rsn];
Costo=[Costo Cost];
if numop>=5 && cont>=p+2
    check=1
    xn=xnbest;
    xr=xrbest;
    rn=rnbest;
    Rsn=Rsnbest;
```

```
        Cost=Costbest;
    end
end
timetaken=toc;      % Tiempo de ejecución del algoritmo

%Gráfica de iteración vs. función objetivo (Confiabilidad) & Costo Total
[AX,H1,H2] = plotyy(conteo,Rs,conteo,Costo,'plot');
set(get(AX(1),'Ylabel'),'String','Confiabilidad de la red (RRT)')
set(get(AX(2),'Ylabel'),'String','Costo Total(C)')
xlabel('Iteración');
title('RRT & C vs. Iteración - Algoritmo de Optimización PLES');
set(H1,'LineStyle','-','Marker','.')
set(H2,'LineStyle','-','Marker','.')
hold on
```

Algoritmo PLES: cálculo estimado de la confiabilidad de la red empleando SMCAD

```
%          Desarrollado y programado por: Paola A. Hernandez-Ramirez
%                   Planteado por:  Noel Artiles-Leon
%                                Mayo 2007
%     Heurístico basado en Programación Lineal Entera Secuencial (PLES)
%
% Función Objetivo:  Maximizar la confiabilidad de una red compleja
% mediante la adición de arcos redundantes con una restricción de
% presupuesto de inversión
%
% Evaluación de la función objetivo aplicando Simulación de Monte Carlo
% empleando la metodología de rutas mínimas (SMCRM)
%
% Variable de entrada:   Vector de máxima redundancia (maxred)
% Variables de salida: * Confiabilidad de la red original (Rso)
%                      * Confiabilidad de la red con arcos redundantes (Rsn)
%                      * Vector de arcos redundantes (xr)
%                      * Tiempo de ejecución del algoritmo (timetaken(seg))
%
% Debe prepararse un archivo .mat que contenga la información general de la
% red compleja a optimizar (matriz de confiabilidades de los arcos de la red
% original (roa), matriz de confiabilidades de los arcos redundantes (rra),
% costo de los arcos (cost) y presupuesto (budget))

function [Rso,Rsn,xr,Cost,timetaken] = PLESsmcad(maxred)

tic;
load DatosRed       % Archivo .mat con la información de la red
maxit=50;           % Número máx. de it. en búsqueda del óptimo
maxbest=10;         % Número máx. de it. después de encontrar un óptimo local
Rso=LSC(ro,sft);    % Límite superior de la confiabilidad - Red Original
% Cálculo de NS en función de Rso y el error esperado
NS=ceil(16*Rso*(1-Rso)*(0.1^(-2)));  % e inicial = 0.1
if NS<10000         % NS mínimo = 100.000 corridas de simulación
    NS=10000;
end
```

```
numa=size(ro,2);    % Número de arcos totales para la red
maxredv=maxred*ones(1,numa);    % Vector con número máximo de arcos
%-------------------------------------------------------------------------%
Xr=[zeros(1,numa);eye(numa);maxredv-x0];    % Matriz de estado inicial
for i=1:size(Xr,1)
    X(i,:)=Xr(i,:)+x0;
end
X1=[ones(size(X,1),1) X];  % Adición de la fila de unos para el intercepto
Y=zeros(size(X,1),1);      % Vector de confiabilidades para c/fila de X
Y(1)=Rso;
for i=2:size(X,1)
    rtemp=1-(1-ro).*(1-rr).^Xr(i,:);
    rtempa=condata(connection,rtemp);
    Y(i)=SMCAD(rtempa,NS);
end
%Coeficientes (betas) de la ecuación de regresión lineal (Y=Bo+B1X1+B2X2..)
betas=regress(Y,X1);
betas1=betas(2:size(betas),1);
lb=x0;                  % Límite inferior del vector solución
ub=x0+ones(1,numa);     % Límite superior del vector solución
for i=1:numa
    if ub(i)>maxredv(i)
        ub(i)=maxredv(i);
    end
end
xint=1:numa;
%Solución inicial de PLE (x: vector de arcos redundantes que Max Rs)
costx0=cost*x0';        % Costo de la red original
budgett=budget+costx0;  % Presupuesto total adicionando el costo de x0
[obj,x]=lp_solve(betas1',cost,budgett,-1,lb,ub,xint);
%-------------------------------------------------------------------------%
xn=x';                      % Nueva red (original + arcos redundantes)
xr=x-x0';                   % Vector de arcos redundantes
rn=1-(1-ro).*(1-rr).^(xr'); % Vector de confiabilidades para la nueva red
rna=condata(connection,rn);
Rsn=SMCAD(rna,NS);          % Confiabilidad de la nueva red
Cost=cost*xr;               % Costo del vector de arcos redundantes
Costt=cost*xn';             % Costo total de la nueva red
% Vectores para gráfica de iteración vs. función objetivo
conteo=1;
Rs=Rsn;
Costo=Cost;
% Vectores de verificación (todas las soluciones generadas sin reemplazo)
Rsver=Rsn;
xrver=xr;
Costver=Cost;
% Inicialización de variables para los criterios de parada
numop=0;
Rsntemp=0;
cont=1;
contbest=0;
check=0;
o=10; p=20; q=30;  % Rangos para ajuste de NS (disminución del error)
Rsnbest=Rsn;
xnbest=xn;
xrbest=xr;
```

```
rnbest=rn;
Costbest=Cost;
while check==0 && cont<maxit && contbest<maxbest
    % Construcción de la matriz de estado: Red solución inicial + un arco i
    % y Red solución inicial - un arco (i)
    Xr=[zeros(1,numa);eye(numa);-eye(numa);maxredv-xn];
    for i=1:size(Xr,1)
        X2(i,:)=Xr(i,:)+xn;
    end
    % Eliminando los vectores que contienen soluciones con arcos que
    % superan el máximo establecido de redundancia, soluciones con arcos
    % negativos y aquellas que eliminan arcos que existen en la red original
    del=zeros(size(X2,1),1);
    for i=1:size(X2,1)
        if isempty(find(X2(i,:)>maxredv))==0 || isempty(find(X2(i,:)<0))==0
            del(i)=1;
        elseif isempty(find(X2(i,:)==0))==0
            ceros=find(X2(i,:)==0);
            if sum(x0(ceros))>0
                del(i)=1;
            end
        end
    end
    Xtemp=zeros(size(X2,1)-sum(del),size(X2,2));
    Xtemp1=Xtemp;
    c=0;
    for i=1:size(del,1)
        if del(i)==0
            Xtemp(i-c,:)=X2(i,:);
            Xtemp1(i-c,:)=Xr(i,:);
        else
            c=c+1;
        end
    end
    X2=Xtemp;
    Xr=Xtemp1;
    for i=1:size(Xr,1)
        Xr(i,:)=xr'+Xr(i,:);
    end
    X3=[ones(size(X2,1),1) X2];
    % Vector de confiabilidades para la nueva matriz de estado
    Y1=zeros(size(Xr,1),1);
    Y1(1)=Rsn;
    for i=2:size(Xr,1)
        rtemp=1-(1-ro).*(1-rr).^Xr(i,:);
        rtempa=condata(connection,rtemp);
        Y1(i)=SMCAD(rtempa,NS);
    end
    % Coeficientes (betas) de la ecuación de regresión lineal
    betas2=regress(Y1,X3);
    betas3=betas2(2:size(betas2),1);
    % Los vectores de límite inferior y superior de las variables de
    % decisión estan sujetos a que puede quitarse un arco sólo si se ha
    % agregado al menos 1 y se puede añadir uno nuevo si no se ha alcanzado
    % el máximo de redundancia establecido (maxred)
    lb=xn-ones(1,numa);     % Límite inferior del vector solución
```

```
    for i=1:numa
        if lb(i)<x0(i)
            lb(i)=x0(i);
        end
    end
    ub=xn+ones(1,numa);     % Límite superior del vector solución
    for i=1:numa
        if ub(i)>maxredv(i)
            ub(i)=maxredv(i);
        end
    end
    xint=1:numa;
    % Nueva solución de PLE
    [obj,x1]=lp_solve(betas3', cost, budgett, -1, lb, ub, xint);
    xr=x1-x0';
    xn=x1';
    rn=1-(1-ro).*(1-rr).^(xr');
    rna=condata(connection,rn);
    Rsn=SMCAD(rna,NS);
    Cost=cost*xr;
    Costt=cost*xn';
%-------------------------------------------------------------------------%
    % Actualización de los vectores de verificación
    Rsver=[Rsver Rsn];
    xrver=[xrver xr];
    Costver=[Costver Cost];
    if Rsn>Rsnbest
        if xr==xrbest
            Rsn=Rsnbest;
            numop=numop+1
        elseif xrbest>=xr
            Rsn=Rsnbest;
            xn=xnbest;
            xr=xrbest;
            rna=rnabest;
            Cost=Costbest;
            numop=numop+1
        else
            contbest=0
            Rsnbest=Rsn
            xnbest=xn;
            xrbest=xr;
            rnabest=rna;
            Costbest=Cost;
            numop=0
        end
    elseif Rsn<Rsnbest
        if xr==xrbest
            Rsn=Rsnbest;
            numop=numop+1;
        elseif xr>=xrbest
            Rsnbest=Rsn
            xnbest=xn;
            xrbest=xr;
            rnabest=rna;
            Costbest=Cost;
```

```
            numop=0
        else
            contbest=contbest+1
            numop=numop+1
            Rsn=Rsnbest;
            xn=xnbest;
            xr=xrbest;
            rna=rnbest;
            Cost=Costbest;
        end
    elseif Rsn==Rsnbest
        if xr==xrbest
            numop=numop+1
        elseif Cost>Costbest
            Rsn=Rsnbest;
            xn=xnbest;
            xr=xrbest;
            rna=rnabest;
            Cost=Costbest;
        end
    end
    if cont==maxit || contbest==maxbest
        xn=xnbest;
        xr=xrbest;
        rna=rnabest;
        Rsn=Rsnbest;
        Cost=Costbest;
    end
    cont=cont+1
    if cont==o
        NS=min(50000,ceil(16*Rsnbest*(1-Rsnbest)*(0.01^(-2)))); % e = 0.01
        Rsnbest=SMCAD(rnabest,NS);
        contbest=0;
        numop=0;
    elseif cont==p
        NS=min(100000,ceil(16*Rsnbest*(1-Rsnbest)*(0.001^(-2)))); % e = 0.001
        Rsnbest=SMCAD(rnabest,NS);
    elseif cont==q
        NS=min(500000,ceil(16*Rsnbest*(1-Rsnbest)*(0.0001^(-2)))); % e =
0.0001
        Rsnbest=SMCAD(rnabest,NS);
    end
    %Vectores para gráfica de iteración vs. función objetivo
    conteo=[conteo cont];
    Rs=[Rs Rsn];
    Costo=[Costo Cost];
    if numop>=5 && cont>=p+2
        check=1
        xn=xnbest;
        xr=xrbest;
        rna=rnabest;
        Rsn=Rsnbest;
        Cost=Costbest;
    end
end
timetaken=toc;      % Tiempo de ejecución del algoritmo
```

```
%Gráfica de iteración vs. función objetivo (Confiabilidad) & Costo Total
[AX,H1,H2] = plotyy(conteo,Rs,conteo,Costo,'plot');
set(get(AX(1),'Ylabel'),'String','Confiabilidad de la red (RRT)')
set(get(AX(2),'Ylabel'),'String','Costo Total(C)')
xlabel('Iteración');
title('RRT & C vs. Iteración - Algoritmo de Optimización PLES');
set(H1,'LineStyle','-','Marker','.')
set(H2,'LineStyle','-','Marker','.')
hold on
```

Algoritmo AG: cálculo estimado de la confiabilidad de la red empleando SMCRM

```
%             Paola A. Hernandez-Ramirez. Mayo de 2007.
%                      Algoritmo Genético (AG)
%
% Función Objetivo:  Maximizar la confiabilidad de una red compleja
% mediante la adición de arcos redundantes con una restricción de
% presupuesto de inversión
%
% Evaluación de la función objetivo aplicando Simulación de Monte Carlo
% empleando la metodología de rutas mínimas (SMCRM)
%
% Variable de entrada:   Vector de máxima redundancia (maxred)
% Variables de salida: * Vector de arcos redundantes (x)
%                      * Confiabilidad de la red con arcos redundantes (Rsn)
%                      * Criterio de parada del algoritmo (reason)
%                      * Tiempo de ejecución del algoritmo (timetaken(seg))
%
% Debe prepararse un archivo .mat que contenga la información general de la
% red compleja a optimizar (Matriz de rutas mínimas (MP), vector de
% confiabilidades de los arcos de la red original (ro), vector de
% confiabilidades de los arcos redundantes (rr), costo de los arcos (cost)
% y presupuesto (budget))

function [x,Rsn,reason,timetaken] = AGsmcrm(maxred)

tic;
load DatosRed   % Archivo .mat con la información de la red
fitnessFcn = @(x) myfunction(x) ;   % Función de aptitud ("fitness")
numberOfVariables = size(ro,2);     % Número de variables
% Límite superior (UB) e inferior (LB) de las variables de decisión
LB = 0*ones(1,numberOfVariables);
UB=[];
for i=1:numberOfVariables
    if x0==0
        UB=[UB maxred];
    else
        UB=[UB maxred-x0(i)];
    end
end
Bound = [LB;UB];
maxredv=maxred*ones(1,size(ro,2));  % Vector con número máximo de arcos
```

```
pc=numberOfVariables*max(maxred);   % Tamaño de la población
% Estructura de opciones para el AG: las opciones llamadas 'CreationFcn',
% 'MutationFcn', y 'PopInitRange' son parte requerida del problema
options =
gaoptimset('CreationFcn',@int_pop,'SelectionFcn',@selectiontournament,...
    'MutationFcn',@int_unif_mutation,'EliteCount',5,'PopInitRange',Bound,...
    'Display','iter','StallGenL',20,'StallTimeL',2400,'Generations',50,...
    'PopulationSize',pc,'CrossoverFraction', 0.5,'PlotFcns',{@gaplotbestfe});
% Aplicación de la función gap (ga modificada) del toolbox de Algoritmos
% Genéticos 1.0.2.
[x, fval, reason, output, population, scores] =
gap(fitnessFcn,numberOfVariables,options);
Rsn=-fval;
timetaken=toc;  % Tiempo de ejecución del algoritmo
%-------------------------------------------------------------------------%
% Función de mutación para generar los hijos satisfaciendo el rango y la
% restricción de variables de decisión enteras
function mutationChildren =
int_unif_mutation(parents,options,GenomeLength,...
    FitnessFcn,state,thisScore,thisPopulation,mutationRate)
if(nargin < 8)
    mutationRate = 0.25;
end
if(strcmpi(options.PopulationType,'doubleVector'))
    mutationChildren = zeros(length(parents),GenomeLength);
    for i=1:length(parents)
        child = thisPopulation(parents(i),:);
        % Cada elemento del cromosoma tiene una prob. del 25% de ser mutado
        mutationPoints = find(rand(1,length(child)) < mutationRate);
        % Cada gen es reemplazado por un valor seleccionado random del rango
        range = options.PopInitRange;
        [r,c] = size(range);
        if(c ~= 1)
            range = range(:,mutationPoints);
        end
        lower = range(1,:);
        upper = range(2,:);
        span = upper - lower;
        % El uso de la función ROUND (redondeo) asegura que los hijos sean
enteros
        child(mutationPoints) = lower + round(rand(1,length(mutationPoints)))
.* span;
        mutationChildren(i,:) = child;
    end
elseif(strcmpi(options.PopulationType,'bitString'))
    mutationChildren = zeros(length(parents),GenomeLength);
    for i=1:length(parents)
        child = thisPopulation(parents(i),:);
        mutationPoints = find(rand(1,length(child)) < mutationRate);
        child(mutationPoints) = ~child(mutationPoints);
        mutationChildren(i,:) = child;
    end
end
% Final de la función de mutación
%-------------------------------------------------------------------------%
% Función para generar la población de posibles vectores solución
```

```
function Population = int_pop(GenomeLength,FitnessFcn,options)
totalpopulation = sum(options.PopulationSize);
range = options.PopInitRange;
lower= range(1,:);
span = range(2,:) - lower;
% El uso de la función ROUND (redondeo) asegura que los hijos sean enteros
Population = repmat(lower,totalpopulation,1) +  ...
    round(repmat(span,totalpopulation,1) .*
rand(totalpopulation,GenomeLength));
% Final de la función de creación
%-------------------------------------------------------------------------%
% Cálculo de la función de aptitud
function RsB=myfunction(x)
load DatosRed
global NS;
% Vector con la confiabilidad de los arcos mejorada si hay redundancia
rtemp=1-(1-ro).*(1-rr).^x;
Cest=x*cost';   % Costo del vector de arcos redundantes
delta=0;
if budget>=Cest
    delta=0;
else
    delta=1;
end
B=delta*((Cest-budget)/budget); % Penalidad por incumplimiento del
presupuesto
Rs=SMCRM(MP,NS,rtemp);  % Cálculo de la confiabilidad de la red con SMCRM
RsB=-Rs+B;              % Función de penalidad (confiabilidad-penalidad)
```

Algoritmo AG: cálculo estimado de la confiabilidad de la red empleando SMCAD

```
%             Paola A. Hernandez-Ramirez. Mayo de 2007.
%                      Algoritmo Genético (AG)
%
% Función Objetivo:  Maximizar la confiabilidad de una red compleja
% mediante la adición de arcos redundantes con una restricción de
% presupuesto de inversión
%
% Evaluación de la función objetivo aplicando Simulación de Monte Carlo
% empleando el algoritmo de Dijkstra (SMCAD)
%
% Variable de entrada:   Vector de máxima redundancia (maxred)
% Variables de salida: * Vector de arcos redundantes (x)
%                      * Confiabilidad de la red con arcos redundantes (Rsn)
%                      * Criterio de parada del algoritmo (reason)
%                      * Tiempo de ejecución del algoritmo (timetaken(seg))
%
% Debe prepararse un archivo .mat que contenga la información general de la
% red compleja a optimizar (matriz de confiabilidades de los arcos de la red
% original (roa), matriz de confiabilidades de los arcos redundantes (rra),
% costo de los arcos (cost) y presupuesto (budget))
```

```
function [x,Rsn,reason,timetaken] = AGsmcad(maxred)

tic;
load DatosRed   % Archivo .mat con la información de la red
fitnessFcn = @(x) myfunction(x) ;   % Función de aptitud ("fitness")
numberOfVariables = size(ro,2);     % Número de variables
% Límite superior (UB) e inferior (LB) de las variables de decisión
LB = 0*ones(1,numberOfVariables);
UB=[];
for i=1:numberOfVariables
    if x0==0
        UB=[UB maxred];
    else
        UB=[UB maxred-x0(i)];
    end
end
Bound = [LB;UB];
maxredv=maxred*ones(1,size(ro,2));  % Vector con número máximo de arcos
pc=numberOfVariables*max(maxred);   % Tamaño de la población
% Estructura de opciones para el AG: las opciones llamadas 'CreationFcn',
% 'MutationFcn', y 'PopInitRange' son parte requerida del problema
options =
gaoptimset('CreationFcn',@int_pop,'SelectionFcn',@selectiontournament,...
    'MutationFcn',@int_unif_mutation,'EliteCount',5,'PopInitRange',Bound,...
    'Display','iter','StallGenL',20,'StallTimeL',2400,'Generations',50,...
    'PopulationSize',pc,'CrossoverFraction', 0.5,'PlotFcns',{@gaplotbestfe});
% Aplicación de la función gap (ga modificada) del toolbox de Algoritmos
% Genéticos 1.0.2.
[x, fval, reason, output, population, scores] =
gap(fitnessFcn,numberOfVariables,options);
Rsn=-fval;
timetaken=toc;  % Tiempo de ejecución del algoritmo
%-------------------------------------------------------------------------%
% Función de mutación para generar los hijos satisfaciendo el rango y la
% restricción de variables de decisión enteras
function mutationChildren =
int_unif_mutation(parents,options,GenomeLength,...
    FitnessFcn,state,thisScore,thisPopulation,mutationRate)
if(nargin < 8)
    mutationRate = 0.25;
end
if(strcmpi(options.PopulationType,'doubleVector'))
    mutationChildren = zeros(length(parents),GenomeLength);
    for i=1:length(parents)
        child = thisPopulation(parents(i),:);
        % Cada elemento del cromosoma tiene una prob. del 25% de ser mutado
        mutationPoints = find(rand(1,length(child)) < mutationRate);
        % Cada gen es reemplazado por un valor seleccionado random del rango
        range = options.PopInitRange;
        [r,c] = size(range);
        if(c ~= 1)
            range = range(:,mutationPoints);
        end
        lower = range(1,:);
        upper = range(2,:);
        span = upper - lower;
```

```
        % El uso de la función ROUND (redondeo) asegura que los hijos sean
enteros
        child(mutationPoints) = lower + round(rand(1,length(mutationPoints)))
.* span;
        mutationChildren(i,:) = child;
    end
elseif(strcmpi(options.PopulationType,'bitString'))
    mutationChildren = zeros(length(parents),GenomeLength);
    for i=1:length(parents)
        child = thisPopulation(parents(i),:);
        mutationPoints = find(rand(1,length(child)) < mutationRate);
        child(mutationPoints) = ~child(mutationPoints);
        mutationChildren(i,:) = child;
    end
end
% Final de la función de mutación
%-------------------------------------------------------------------------%
% Función para generar la población de posibles vectores solución
function Population = int_pop(GenomeLength,FitnessFcn,options)
totalpopulation = sum(options.PopulationSize);
range = options.PopInitRange;
lower= range(1,:);
span = range(2,:) - lower;
% El uso de la función ROUND (redondeo) asegura que los hijos sean enteros
Population = repmat(lower,totalpopulation,1) +  ...
    round(repmat(span,totalpopulation,1) .*
rand(totalpopulation,GenomeLength));
% Final de la función de creación
%-------------------------------------------------------------------------%
% Cálculo de la función de aptitud
function RsB=myfunction(x)
load DatosRed
global NS;
% Vector con la confiabilidad de los arcos mejorada si hay redundancia
rtemp=1-(1-ro).*(1-rr).^x;
rtempm=condata(connection,rtemp); % Conversión a matriz de confiabilidades
Cest=x*cost';   % Costo del vector de arcos redundantes
delta=0;
if budget>=Cest
    delta=0;
else
    delta=1;
end
B=delta*((Cest-budget)/budget); % Penalidad por incumplimiento del
presupuesto
Rs=SMCAD(rtempm,NS);    % Cálculo de la confiabilidad de la red con SMCAD
RsB=-Rs+B;              % Función de penalidad (confiabilidad-penalidad)
```

Apéndice B. Resultados de Experimentos

Resultados del primer experimento factorial – comparación algoritmos de optimización (A) y metodologías para estimar la confiabilidad de la red (MC)

Réplica	Tamaño de Red (*n*)	Algoritmo (*A*)		Método de confiabilidad (*MC*)		Respuestas	
		AG	PLES	SMCRM	SMCAD	R_{RT}	t_e
1	6	*		*		0.5619	19.23
2	6	*		*		0.3528	14.01
3	6	*		*		0.5311	19.54
4	6	*		*		0.5941	18.84
5	6	*		*		0.6563	11.81
1	9	*		*		0.7677	61.65
2	9	*		*		0.6296	32.98
3	9	*		*		0.6199	91.54
4	9	*		*		0.6719	83.78
5	9	*		*		0.6861	88.32
1	12	*		*		0.7058	1216.43
2	12	*		*		0.5705	1182.85
3	12	*		*		0.8912	827.53
4	12	*		*		0.8854	866.06
5	12	*		*		0.8078	853.91
1	6		*	*		0.5612	10.55
2	6		*	*		0.3518	8.66
3	6		*	*		0.5319	9.3
4	6		*	*		0.5917	10.8
5	6		*	*		0.6563	11.02
1	9		*	*		0.7532	38.7
2	9		*	*		0.6298	32.98
3	9		*	*		0.6215	52.67
4	9		*	*		0.6606	30.04
5	9		*	*		0.6865	53.6
1	12		*	*		0.6989	768.54
2	12		*	*		0.5684	730.12
3	12		*	*		0.8854	448.26
4	12		*	*		0.8854	485.88
5	12		*	*		0.7981	852.3

Réplica	Tamaño de Red (*n*)	Algoritmo (*A*)		Método de confiabilidad (*MC*)		Respuestas	
		AG	PLES	SMCRM	SMCAD	R_{RT}	t_e
1	6	*			*	0.5621	42.68
2	6	*			*	0.3514	38.25
3	6	*			*	0.5321	46.07
4	6	*			*	0.5914	39.97
5	6	*			*	0.6517	37.56
1	9	*			*	0.7693	200.58
2	9	*			*	0.6425	183.5
3	9	*			*	0.6237	146.3
4	9	*			*	0.6705	147.85
5	9	*			*	0.6844	184.16
1	12	*			*	0.7027	790.92
2	12	*			*	0.5684	993.51
3	12	*			*	0.8847	836.87
4	12	*			*	0.8885	636.66
5	12	*			*	0.8045	980.29
1	6		*		*	0.5619	38.71
2	6		*		*	0.3518	24.09
3	6		*		*	0.5318	27.97
4	6		*		*	0.5917	32.58
5	6		*		*	0.6572	26.71
1	9		*		*	0.7663	174.4
2	9		*		*	0.6428	135.56
3	9		*		*	0.6256	133.74
4	9		*		*	0.672	114.44
5	9		*		*	0.6865	148.59
1	12		*		*	0.6982	624.26
2	12		*		*	0.5684	821.89
3	12		*		*	0.8854	424.98
4	12		*		*	0.8854	475.64
5	12		*		*	0.8078	753.91

Resultados del segundo experimento factorial – comparación algoritmos de optimización (A) empleando SMCAD para estimar la confiabilidad de la red

Réplica	Tamaño de Red (*n*)	Algoritmo (*A*)		Respuestas	
		AG	PLES	R_{RT}	t_e
1	6	*		0.5621	42.68
2	6	*		0.3514	38.25
3	6	*		0.5321	46.07
4	6	*		0.5914	39.97
5	6	*		0.6517	37.56
1	9	*		0.7693	200.58
2	9	*		0.6425	183.5
3	9	*		0.6237	146.3
4	9	*		0.6705	147.85
5	9	*		0.6844	184.16
1	12	*		0.7027	790.92
2	12	*		0.5684	993.51
3	12	*		0.8847	836.87
4	12	*		0.8885	636.66
5	12	*		0.8045	980.29
1	16	*		0.8145	2505.8
2	16	*		0.7755	3064.55
3	16	*		0.9553	2333.66
4	16	*		0.936	2382.21
5	16	*		0.9068	1764.72
1	20	*		0.9369	6209.6
2	20	*		0.9406	5557.56
3	20	*		0.9819	6053.7
4	20	*		0.9675	6613.91
5	20	*		0.9528	5045.07
1	6		*	0.5619	38.71
2	6		*	0.3518	24.09
3	6		*	0.5318	27.97
4	6		*	0.5917	32.58
5	6		*	0.6572	26.71
1	9		*	0.7663	174.4
2	9		*	0.6428	135.56
3	9		*	0.6256	133.74
4	9		*	0.672	114.44
5	9		*	0.6865	148.59

Réplica	Tamaño de Red (*n*)	Algoritmo (*A*)		Respuestas	
		AG	PLES	R_{RT}	t_e
1	12		*	0.6982	624.26
2	12		*	0.5684	821.89
3	12		*	0.8854	424.98
4	12		*	0.8854	475.64
5	12		*	0.8078	753.91
1	16		*	0.8145	877.26
2	16		*	0.7888	1198.64
3	16		*	0.9457	909.41
4	16		*	0.9388	875.49
5	16		*	0.8992	676.65
1	20		*	0.9293	2982.5
2	20		*	0.9439	2646.4
3	20		*	0.9819	2401.45
4	20		*	0.9591	2771.97
5	20		*	0.9682	2532.64